

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER SYRU/DECE/TR-84/3	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) SOLUTION OF SOME ADDITIONAL ELECTROMAGNETIC PROBLEMS BY THE DISCRETE CONVOLUTION METHOD		5. TYPE OF REPORT & PERIOD COVERED Technical Report No. 23
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Htay L. Nyo Roger F. Harrington		8. CONTRACT OR GRANT NUMBER(s) N00014-76-C-0225
9. PERFORMING ORGANIZATION NAME AND ADDRESS Dept. of Electrical & Computer Engineering Syracuse University Syracuse, New York 13210		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Department of the Navy Office of Naval Research Arlington, Virginia 22217		12. REPORT DATE February 1984
		13. NUMBER OF PAGES 77
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Computer Program Discrete Convolution Large Matrices Method of Moments		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This report gives solution results and computer programs for the Discrete Convolution Method applied to scattering from a helix, to radiation from planar array antennas with antenna elements arranged in triangular patterns (solved using one expansion function per element), and to radiation from planar array antennas with antenna elements arranged in triangular patterns (solved using three expansion functions per element). It also gives a brief discussion on the spatial domain interpretation of the spectral domain iteration technique.		

DD FORM 1 JAN 73 1473

EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

SYRU/DECE/TR-84/3

SOLUTION OF SOME ADDITIONAL ELECTROMAGNETIC PROBLEMS
BY THE DISCRETE CONVOLUTION METHOD

by

Htay L. Nyo
Roger F. Harrington

Department of
Electrical and Computer Engineering
Syracuse University
Syracuse, New York 13210

Technical Report No. 23
February 1984

Contract No. N00014-76-C-0225

Approved for public release; distribution unlimited

Reproduction in whole or in part permitted for any
purpose of the United States Government.

Prepared for

DEPARTMENT OF THE NAVY
OFFICE OF NAVAL RESEARCH
ARLINGTON, VIRGINIA 22217

SYRU/DECE/TR-84/3

SOLUTION OF SOME ADDITIONAL ELECTROMAGNETIC PROBLEMS
BY THE DISCRETE CONVOLUTION METHOD

by

Htay L. Nyo
Roger F. Harrington

Department of
Electrical and Computer Engineering
Syracuse University
Syracuse, New York 13210

Technical Report No. 23
February 1984

Contract No. N00014-76-C-0225

Approved for public release; distribution unlimited

Reproduction in whole or in part permitted for any
purpose of the United States Government.

Prepared for

DEPARTMENT OF THE NAVY
OFFICE OF NAVAL RESEARCH
ARLINGTON, VIRGINIA 22217

CONTENTS

	Page
I. INTRODUCTION-----	1
II. SAMPLE COMPUTATIONS AND COMMENTS-----	1
III. CONCLUSIONS-----	33
APPENDIX	
A. SPATIAL DOMAIN INTEPRETATION OF THE SPECTRAL ITERATION TECHNIQUES-----	35
B. COMPUTER PROGRAMS-----	43
I. MAIN PROGRAM SEGMENT AND SUBROUTINES FOR THE HELIX PROBLEM-----	43
II. MAIN PROGRAM SEGMENT AND SUBROUTINES FOR THE PLANAR ARRAY PROBLEM (SINGLE EXPANSION FUNCTION PER ANTENNA ELEMENT)-----	50
III. MAIN PROGRAM SEGMENT AND SUBROUTINES FOR THE PLANAR ARRAY PROBLEMS (THREE EXPANSION FUNCTIONS PER ANTENNA ELEMENT)-----	58
REFERENCES-----	74

I. INTRODUCTION

In the first report [1] on discrete convolution method for solving some large moment equations, three types of one and two dimensional problems were solved. In this report, we solve three more types of one and two dimensional problems. They are

- (i) scattering from a helix
- (ii) planar arrays with antenna elements arranged in triangular pattern, solved using one expansion function per element
- (iii) planar arrays with antenna elements arranged in triangular pattern, solved using three expansion functions per element

As in the first report, the computing time measurements (made on a KL/10 machine) and the number of iterations needed for the given accuracy are listed. In addition, the array factor of the planar arrays are given.

II. SAMPLE COMPUTATIONS AND COMMENTS

The "one" dimensional problem is scattering from a helix. Fig. 1 shows the problem of scattering from a helix. The MOM formulation of the helix problem requires that the helix be subsectioned into equal length segments. If we number the helix segments so that the segment numbers are in consecutive increasing order from top to bottom, then it is

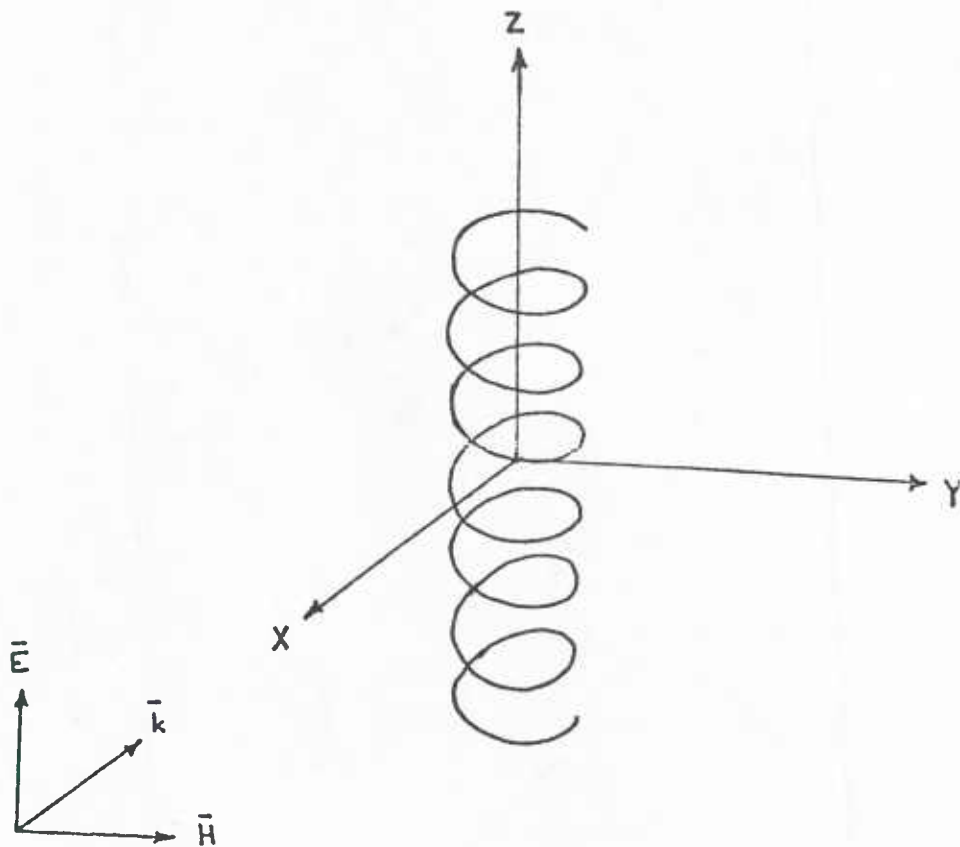


Fig. 1. The helical wire scatterer

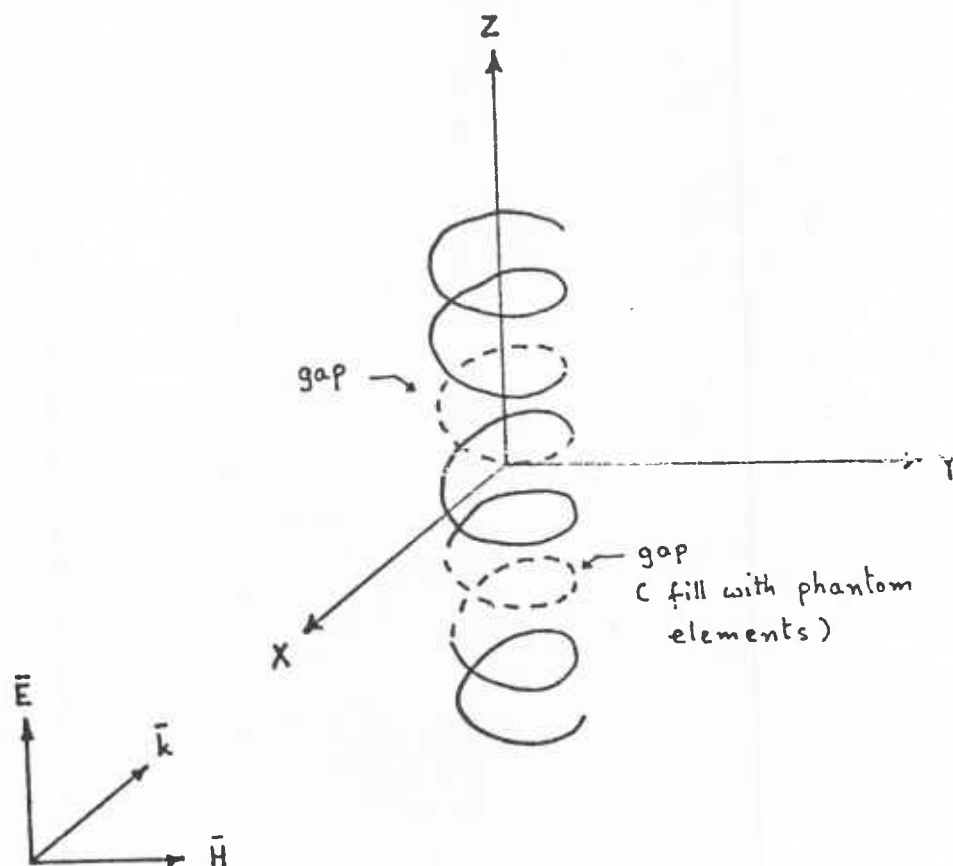


Fig. 2. The helical wire scatterer with gaps

easy to see that the mutual coupling between segments as given by the MOM impedance matrix Z is

$$Z_{mn} = Z(m-n) \quad (1)$$

i.e., the value is dependent only on the difference between segment numbers. Therefore the Z matrix will be Toeplitz if the helix is a complete helix as shown in Fig. 1. The Z matrix will be non-Toeplitz if the helix has gaps in between as shown in Fig. 2. It is apparent from the discussions of other one dimensional problems that both problems can be solved using the one dimensional DCM technique. For the helix with gaps all we need is to insert phantom elements as shown in Fig. 2.

Table 1 gives the number of iterations needed to get the required degree of accuracy for the helix problem. We can see that the number of iterations needed is practically independent of the length of the helix.

The problem of a planar array with antennas arranged in triangular patterns instead of rectangular, can also be formulated as a two dimensional convolution equation by adding phantom elements (as shown in Fig. 4), to make a parallelogram. The triangular pattern arrangement is shown in Fig. 3.

The MOM formulation using one expansion per antenna then gives a block Toeplitz matrix which can be solved using two dimensional DCM. However, it cannot be solved using the block Toeplitz method since the field on the phantom

Table 1. Results for some helical scatterer problems

Radius	Pitch	Number of turns	Number of segments	Number of iterations	Field error (%)	Last current change (%)
.125	.30	6	120	22	.693	.823
					.0498	.1
.125	.30	6	120	23*	1.01	1.11
					.141	.113
.15	.50	6	120	8	.812	1.39
					.0504	.202
				11	.0562	.0958
					.0036	.0142
.15	.30	6	120	10	1.56	1.17
					.177	.248
				16	.0668	.0496
					.0076	.0109
.15	.30	8	160	12	.568	1.21
					.0375	.0778
				16	.0697	.153
					.0046	.0097
.15	.30	8	160	13*	.453	.883
					.0572	.0467
				16*	.0934	.184
					.0118	.0097

Table 1. continued

Radius	Pitch	Number of turns	Number of segments	Number of iterations	Field error (%)	Last current change (%)
.15	.30	8	160	11*	1.07	.785
					.104	.141
				16*	.0821	.0602
					.0082	.0114
.15	.30	16	320	13	.395	.659
					.0154	.0295
				16	.0848	.139
					.0033	.00634
.25	.50	8	160	11	.537	1.86
					.0282	.2
				16	.0495	.163
					.0026	.0184

Here, Radius is the radius of the helix in wavelengths

Pitch is the pitch of the helix in wavelengths

the wire radius in all cases is .006 wavelengths

first entries for each problem in the last two columns

are maximum values and second entries are average values

indicates that polarity of impressed E field is not as shown in Fig. 10 but is z directed

* indicates that the impressed field is not from the direction shown in Fig. 10 but is from z direction



Fig. 3. The triangular pattern arrangement.

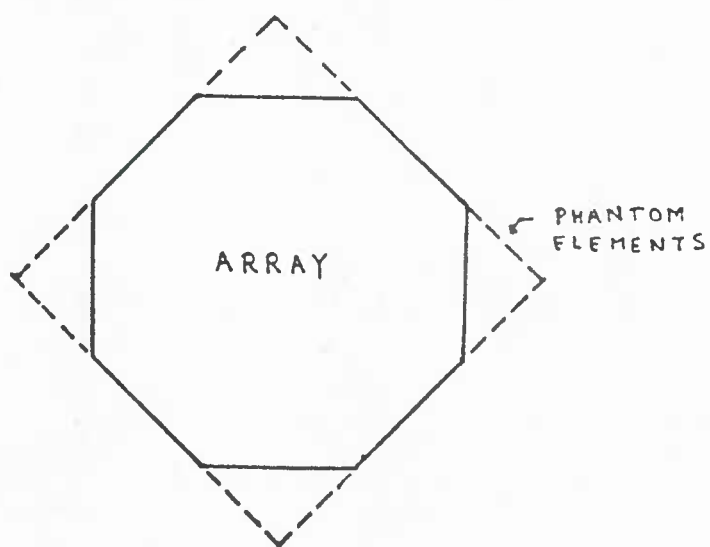


Fig. 4. A planar array with antennas arranged in triangular pattern.

elements are unknown. . Therefore, only LU decomposition or two dimensional DCM can be used. For a large array, DCM will be considerably faster. Since the current on each antenna is not symmetric, using three expansion functions per antenna gives a much more accurate result and needs five times more computing time for the DCM. With LU decomposition method computing time will go up twenty seven times the already large value.

Table 2 lists the computing time and number of iterations needed for the DCM solution using one expansion function per antenna element. Table 3 lists the computing time and number of iterations needed for the DCM solution using three expansion functions per antenna element. All the problems are for planar arrays with 0.48 wavelength antennas one-quarter wavelength in front of the infinite ground plane. The separation between antennas is one-half wavelength in either direction.

The graphs given in Figs. 6, 7, 8, 9, 10, and 11 are the array factors for the planar arrays with triangular pattern arrangement solved by the DCM using one expansion function per antenna element. The array factors are computed in the plane perpendicularly bisecting the array as shown in Fig. 5. Angle measurements are as shown. In all the figures, the solid lines give the array factors for the solutions which take the mutual coupling between antennas into account and the dashed lines are for the idealized solutions which do not take the mutual coupling into

Table 2. Results for some planar arrays with triangular pattern arrangement

N	Excitation	I	Computing	Field Error	Current
			Time(secs)	(%)	Change(%)
12	Uniform	5	4	.1789	.664
				.0867	.378
	Beam steer (45°)	6	4	.1124	.595
				.0554	.173
76	Uniform	6	20	.1447	.489
				.0283	.116
	Beam steer (45°)	6	20	.2768	.796
				.0333	.098
372	Uniform	5	105	.4950	1.674
				.0467	.174
	Beam steer (45°)	6	105	.3636	.863
				.0163	.0456

Here, N is the number of antennas in the array

I is the number of iterations needed to get the given accuracy. For both field error and (last) current change, the upper entry is the maximum and the lower entry is the average.

Table 3. Results for some planar arrays with triangular pattern arrangement (Multiple Expansion Solutions)

N	Excitation	I	Computing	Field Error	Current
			Time(secs)	(%)	Change(%)
76	Uniform	6	67	.0401	.3612
				.0027	.0836
	Beam steer (xz 45°)	6	67	.078	.6898
				.0032	.0704
	Beam steer (yz 135°)	6	67	.0251	.527
				.0016	.0493
372	Uniform	5	165	.1450	1.298
				.0048	.134
	Beam steer (xz 45°)	5	165	.2928	2.1100
				.0047	.0994
	Beam steer (yz 135°)	5	165	.0706	1.43
				.0029	.0748

Here, N is the number of antennas in the array

I is the number of iterations needed to get the given accuracy. For both field error and (last) current change, the upper entry is the maximum and the lower entry is the average.

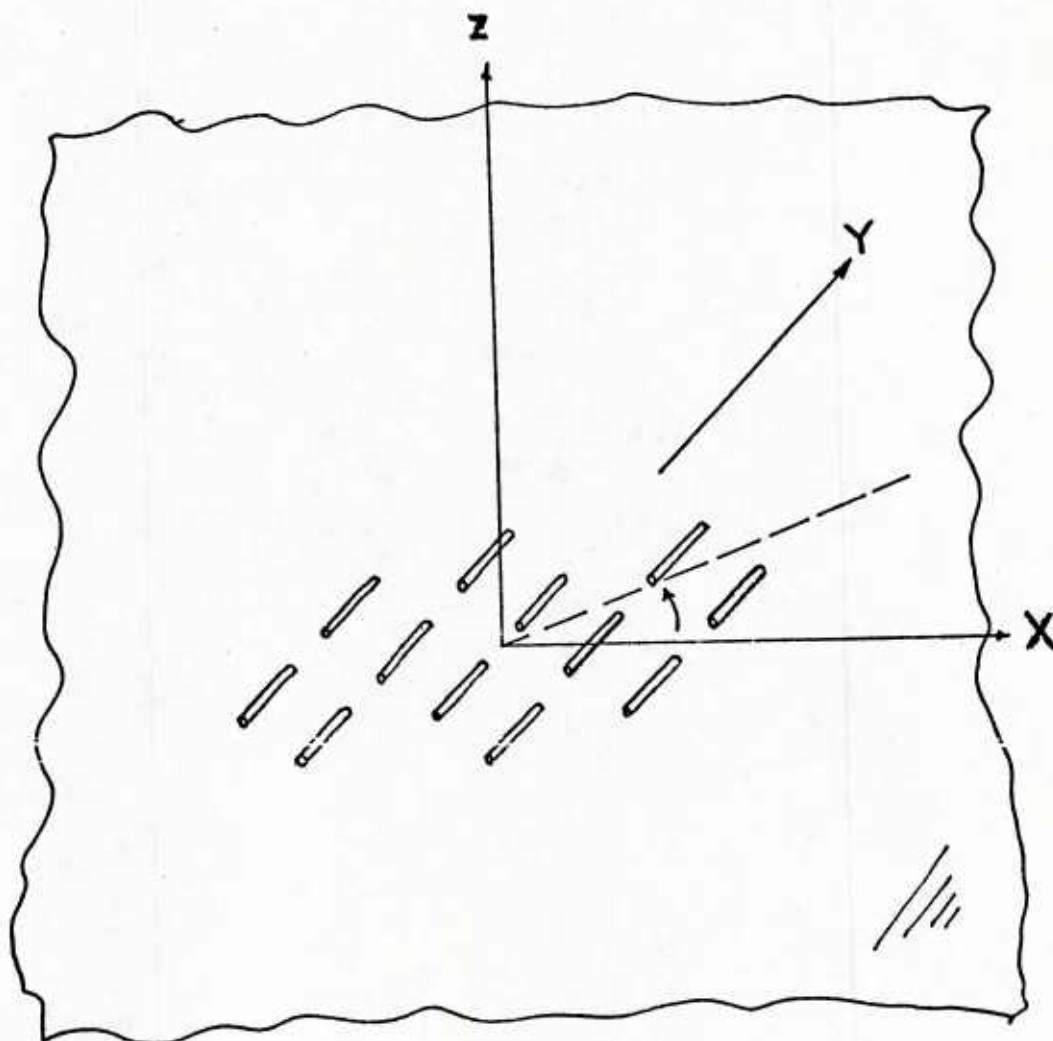


Fig. 5. The relative position of the plane in which the array factors are computed.

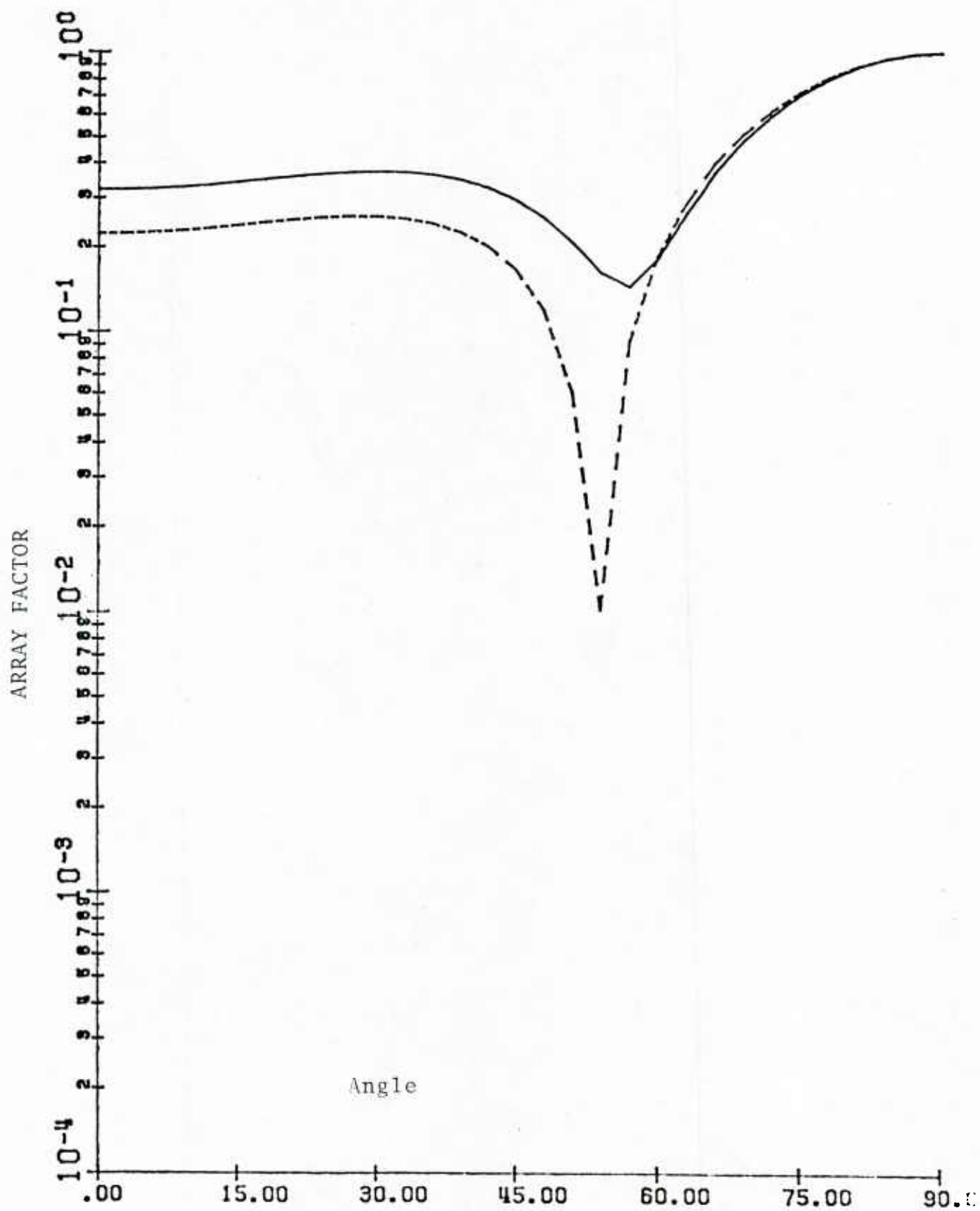


Fig. 6 Array factor of the twelve antenna planar array with uniform excitation

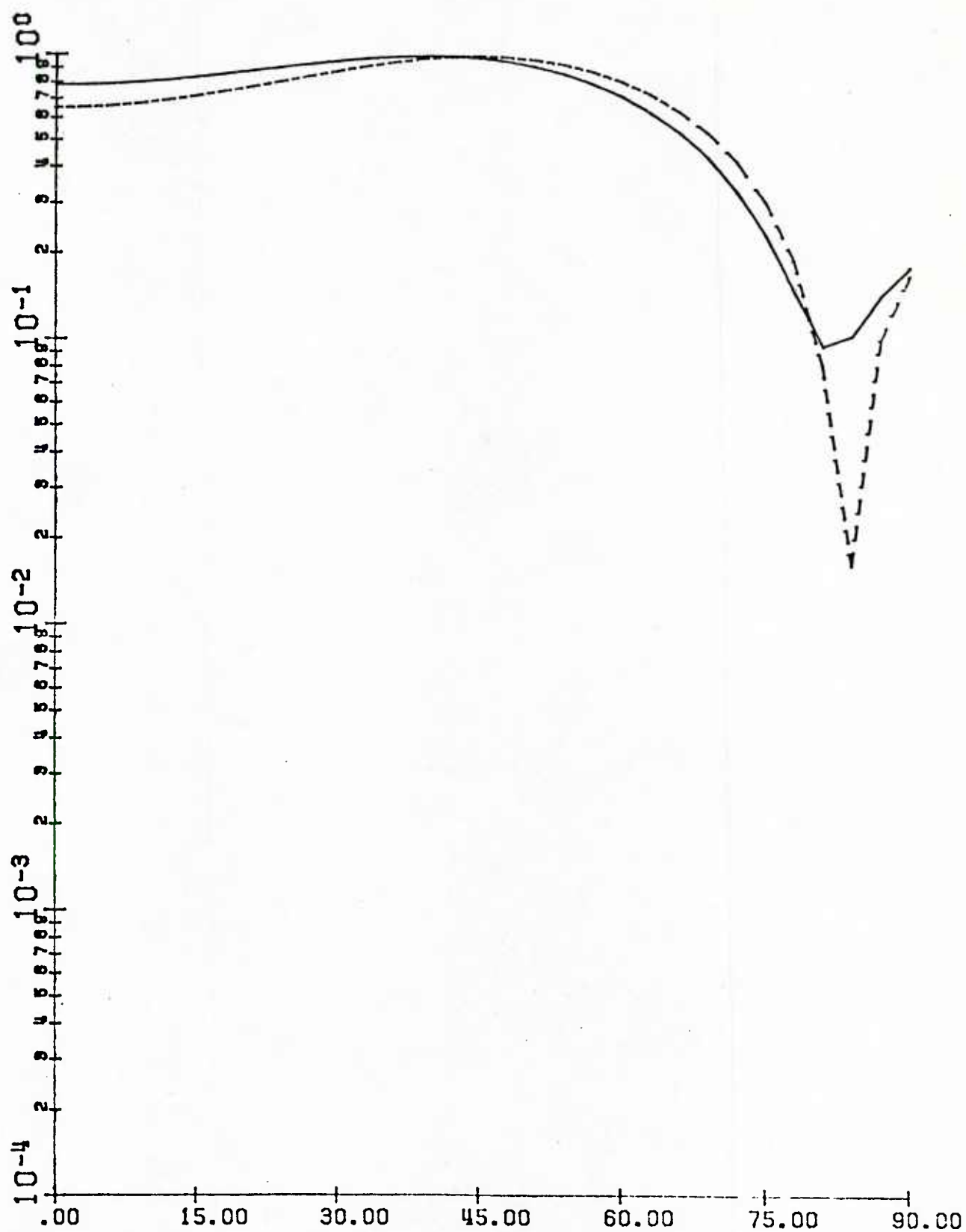


Fig. 7 Array factor of the twelve antenna planar array with excitation to give a 45 degrees scan

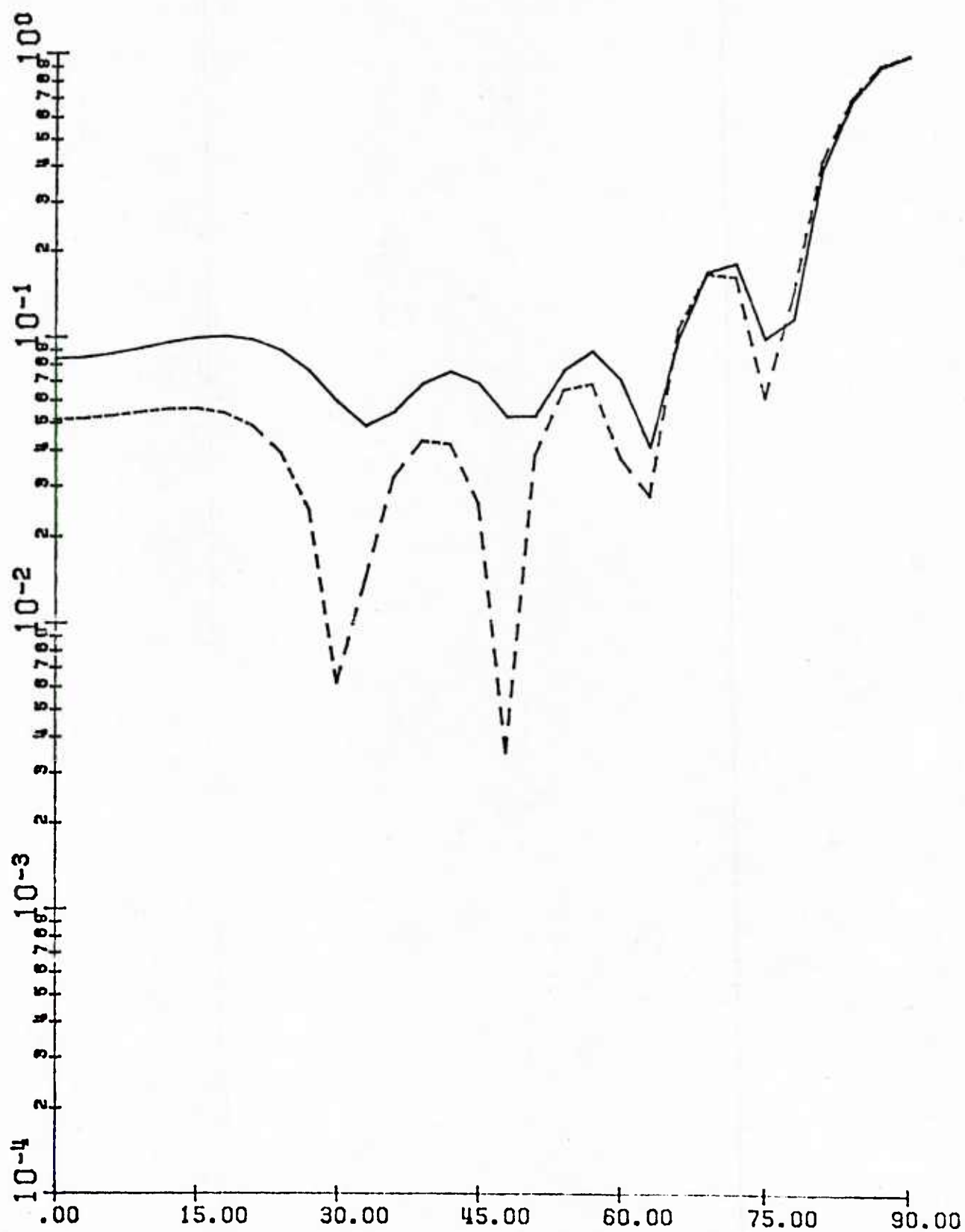


Fig. 8 Array factor of the seventy six antenna planar array with uniform excitation

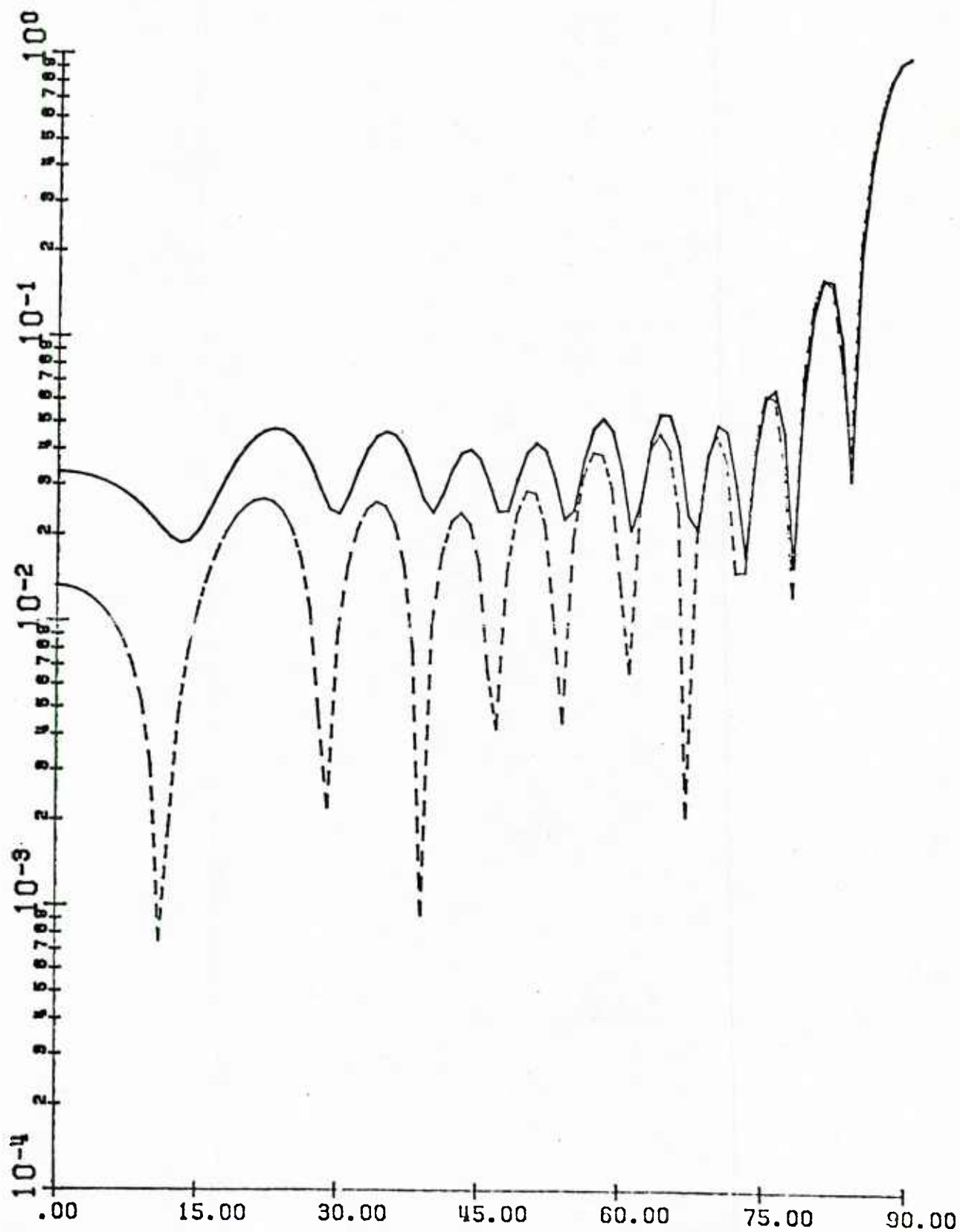


Fig. 10 Array factor of the three hundred and seventy two antenna planar array with uniform excitation

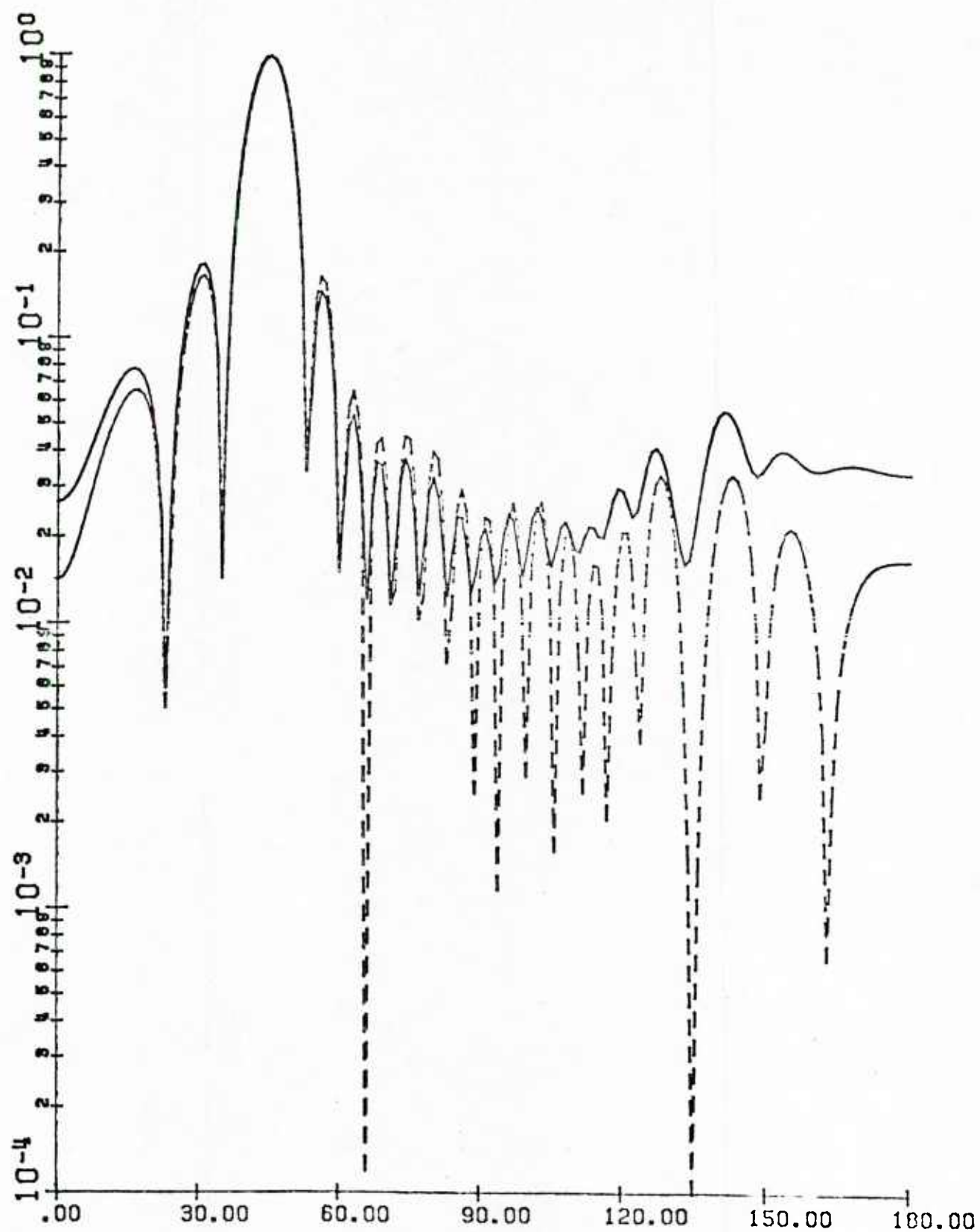


Fig. 11 Array factor of the three hundred and seventy two antenna planar array with excitation to give a 45 degrees scan

account.

As we can see from the graphs, for larger arrays, the main beam is not effected by the mutual coupling but the side lobes and the nulls are effected strongly.

The graphs given in Figs. 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, and 24 are the arrays factors for the planar arrays with triangular pattern arrangement solved by the DCM using three expansion functions per antenna element. The array factors are computed in the xz and yz planes perpendicularly bisecting the array as shown in Fig. 12. Angle measurements are as shown. In all the figures, the solid lines give the array factors for the solutions which takes the mutual coupling between antennas into account and the dashed lines are for the idealized solutions which do not take the mutual coupling into account.

The array factor is defined as the far field pattern divided by the element factor. Therefore for the DCM solution using one expansion function per element, the element factor is the far field pattern of the expansion function and so the array factor can be and is computed directly from the solved currents. However, with three expansions per antenna element, the current distribution over each element is different from the others. Therefore, since our main purpose in computing array factors is to compare the far field patterns, we define a normalized array factor as the total far field pattern divided by the far

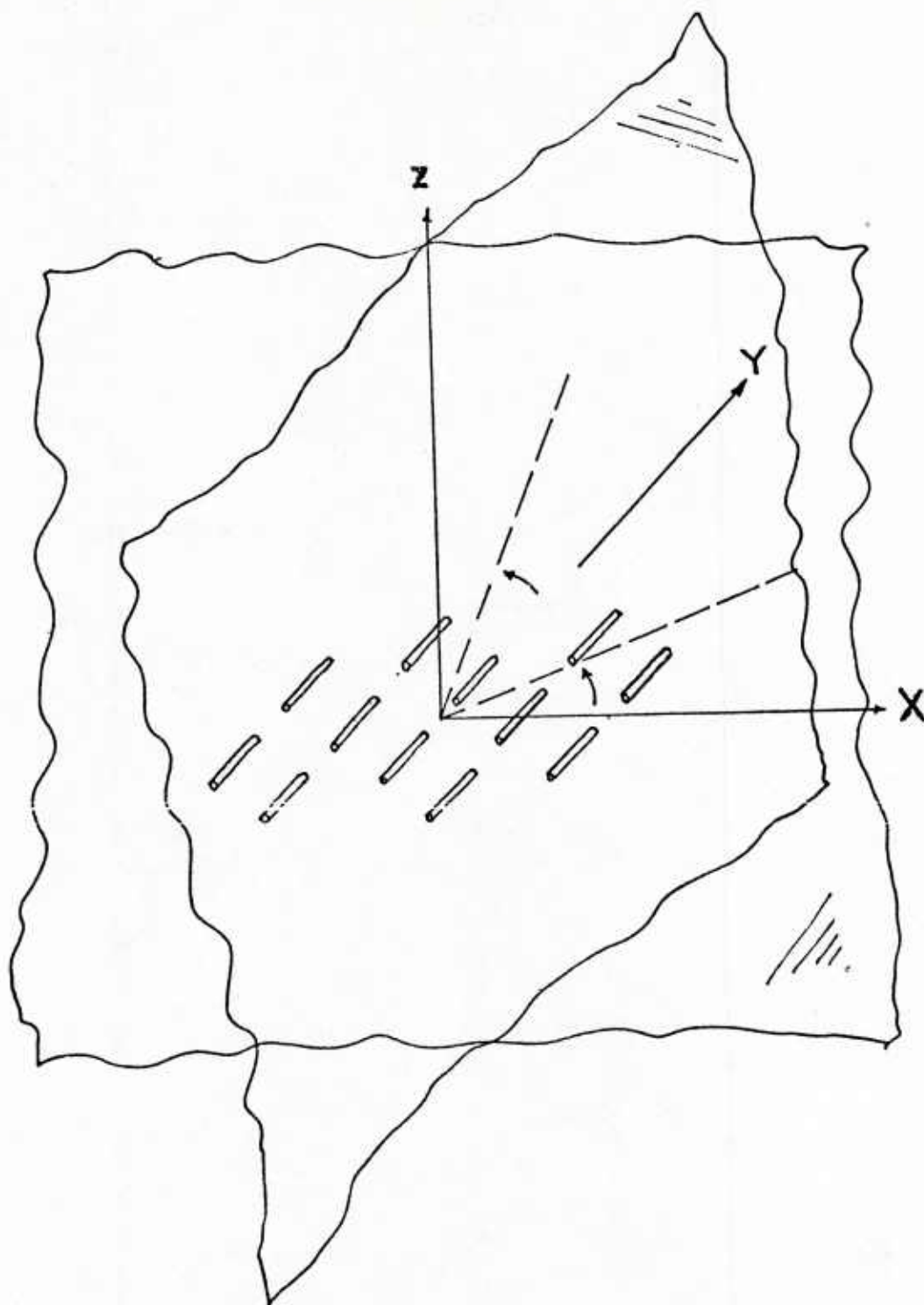


Fig. 12. The relative position of the planes
in which the array factors are computed.

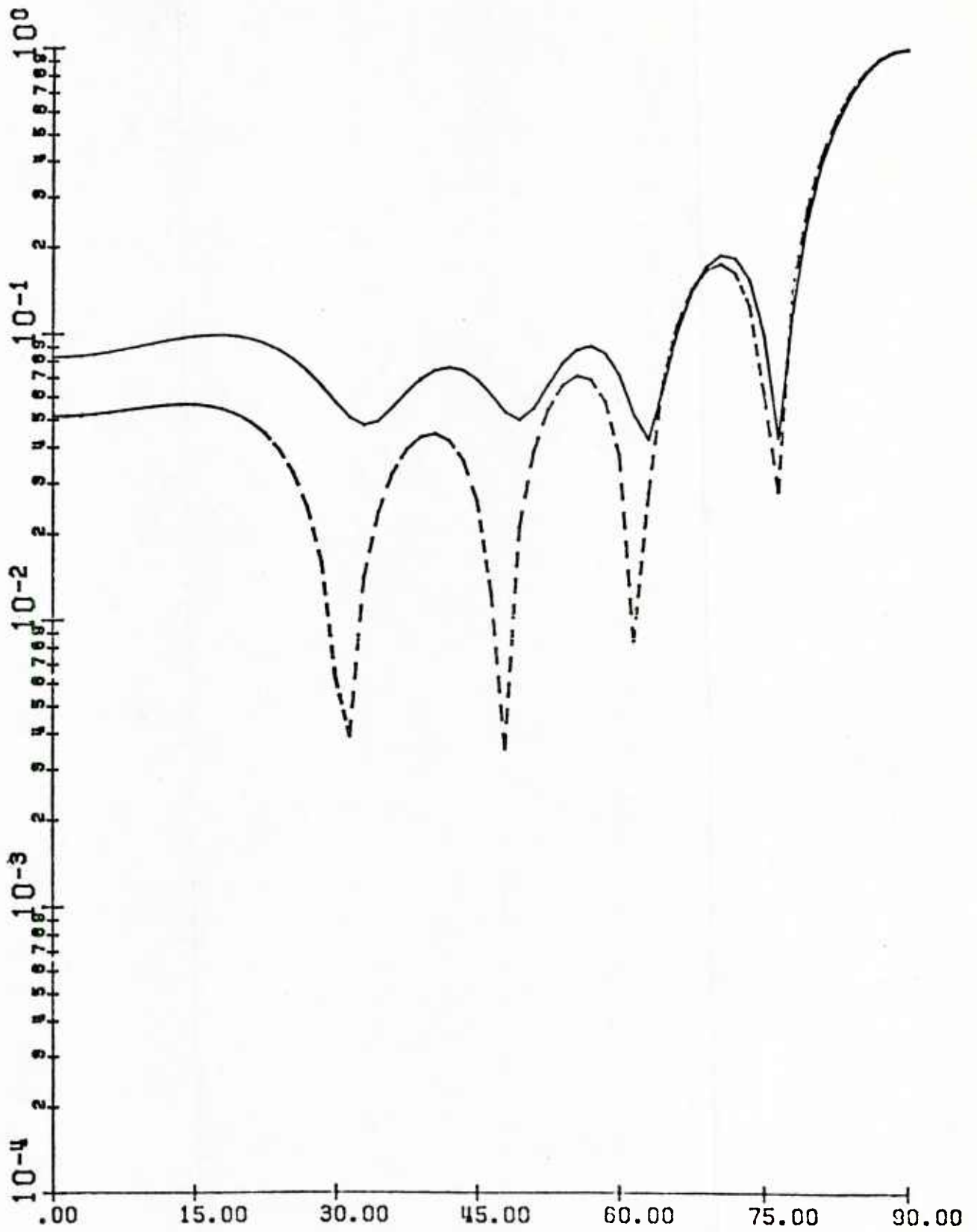


Fig. 13 Array factor in the xz plane of the seventy six antenna planar array with uniform excitation

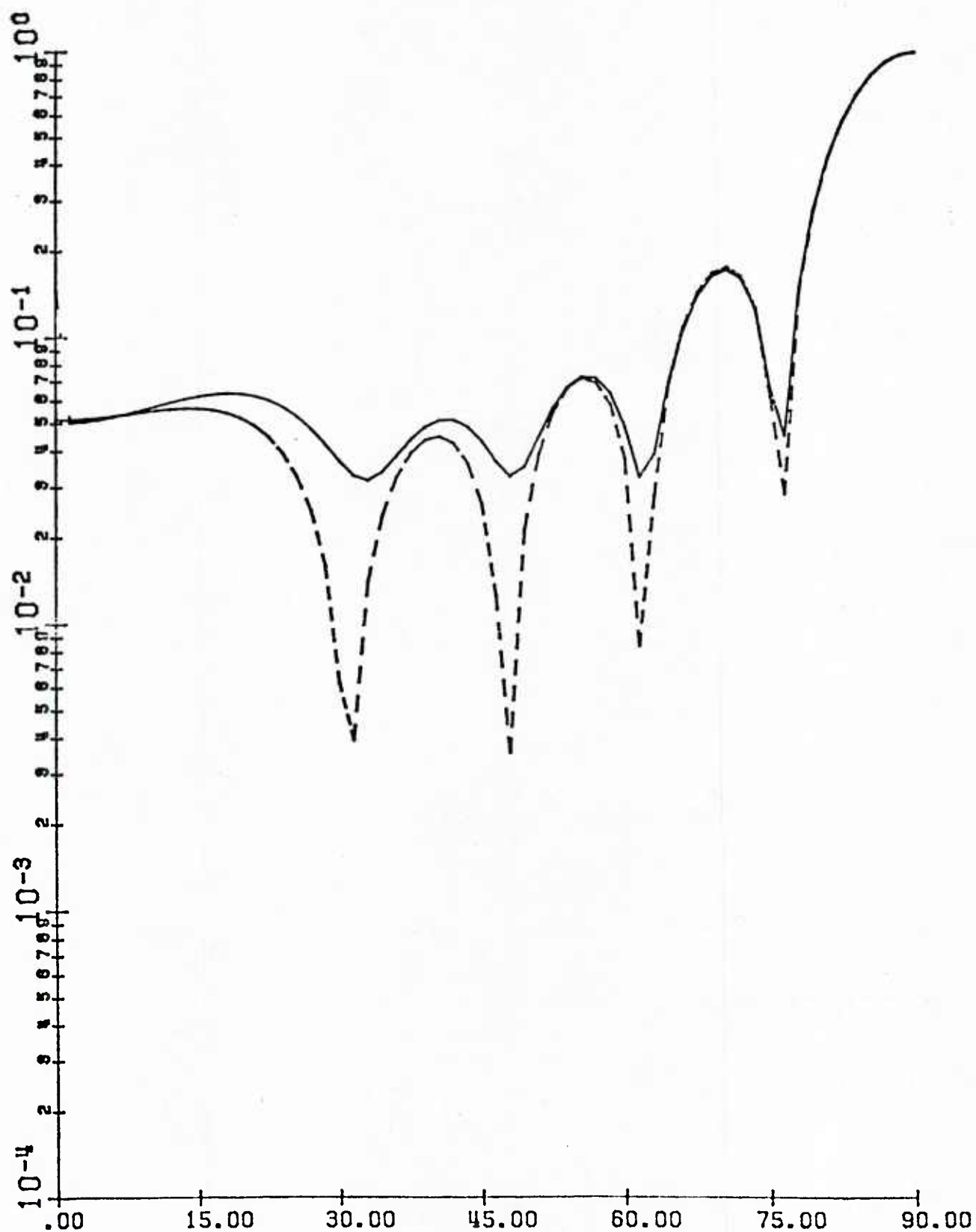


Fig. 14 Array factor in the yz plane of the seventy six antenna planar array with uniform excitation

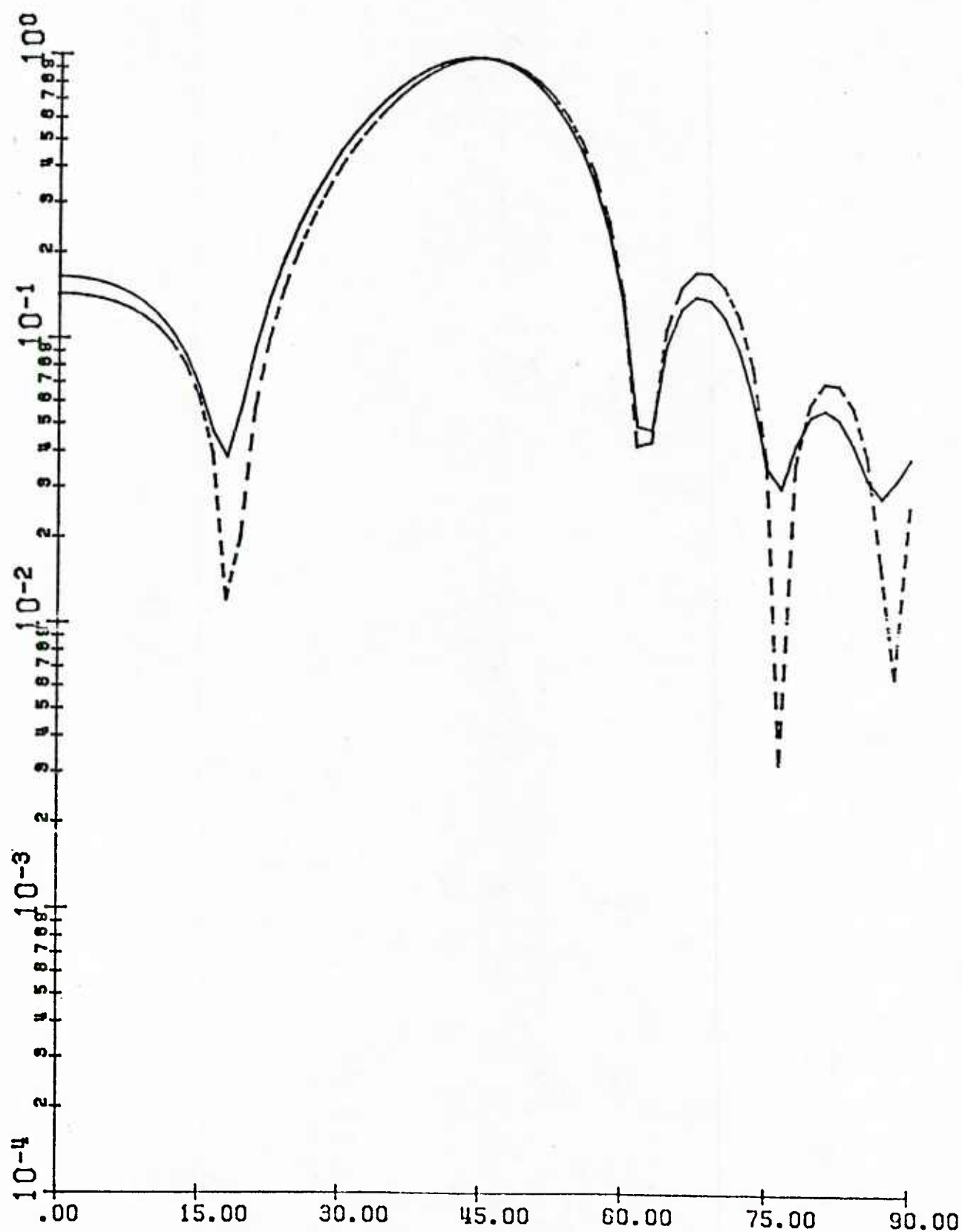


Fig. 15 Array factor in the xz plane of the seventy six antenna planar array with excitation to give a 45 degrees scan in the xz plane

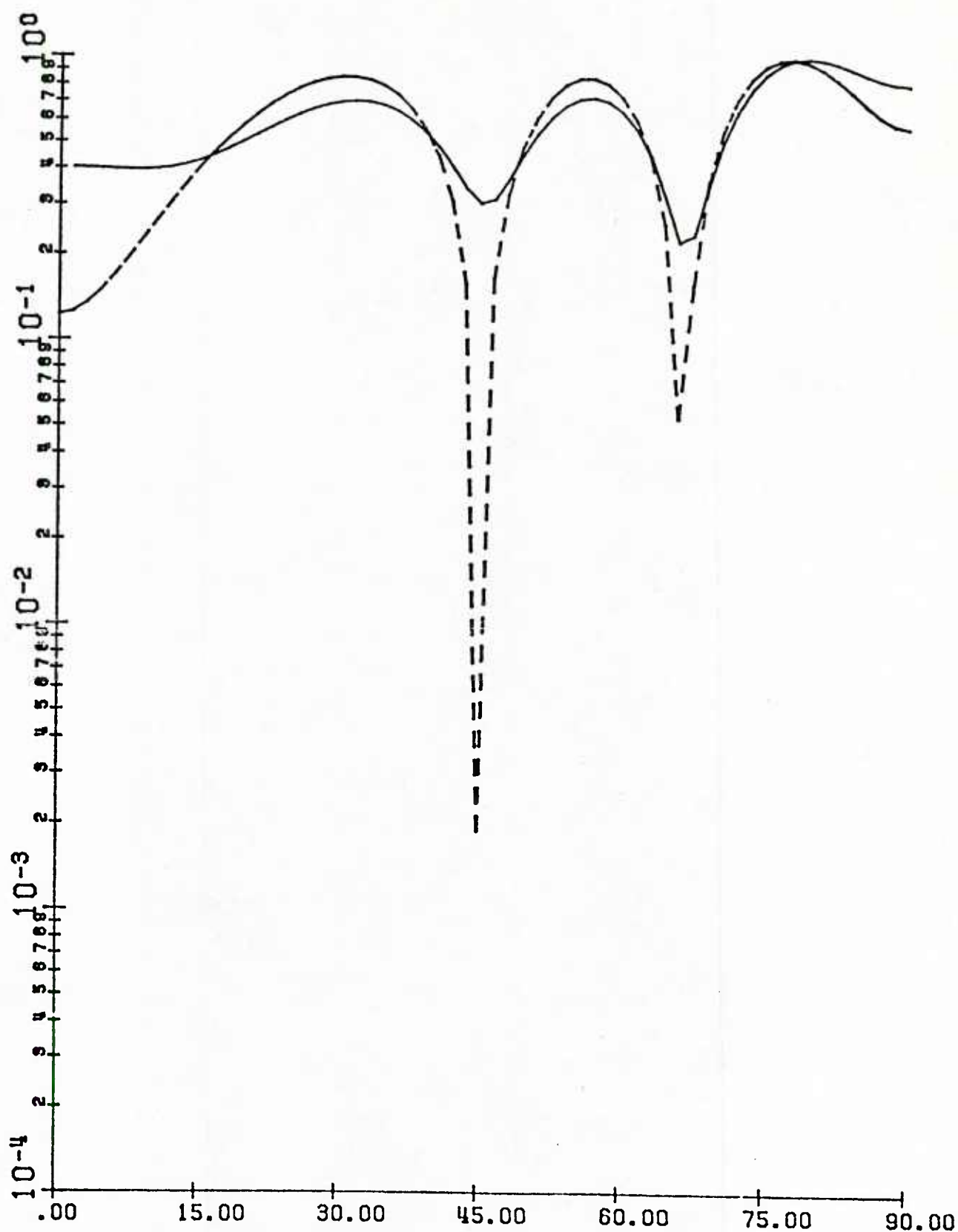


Fig. 16 Array factor in the yz plane of the seventy six antenna planar array with excitation to give a 45 degrees scan in the xz plane

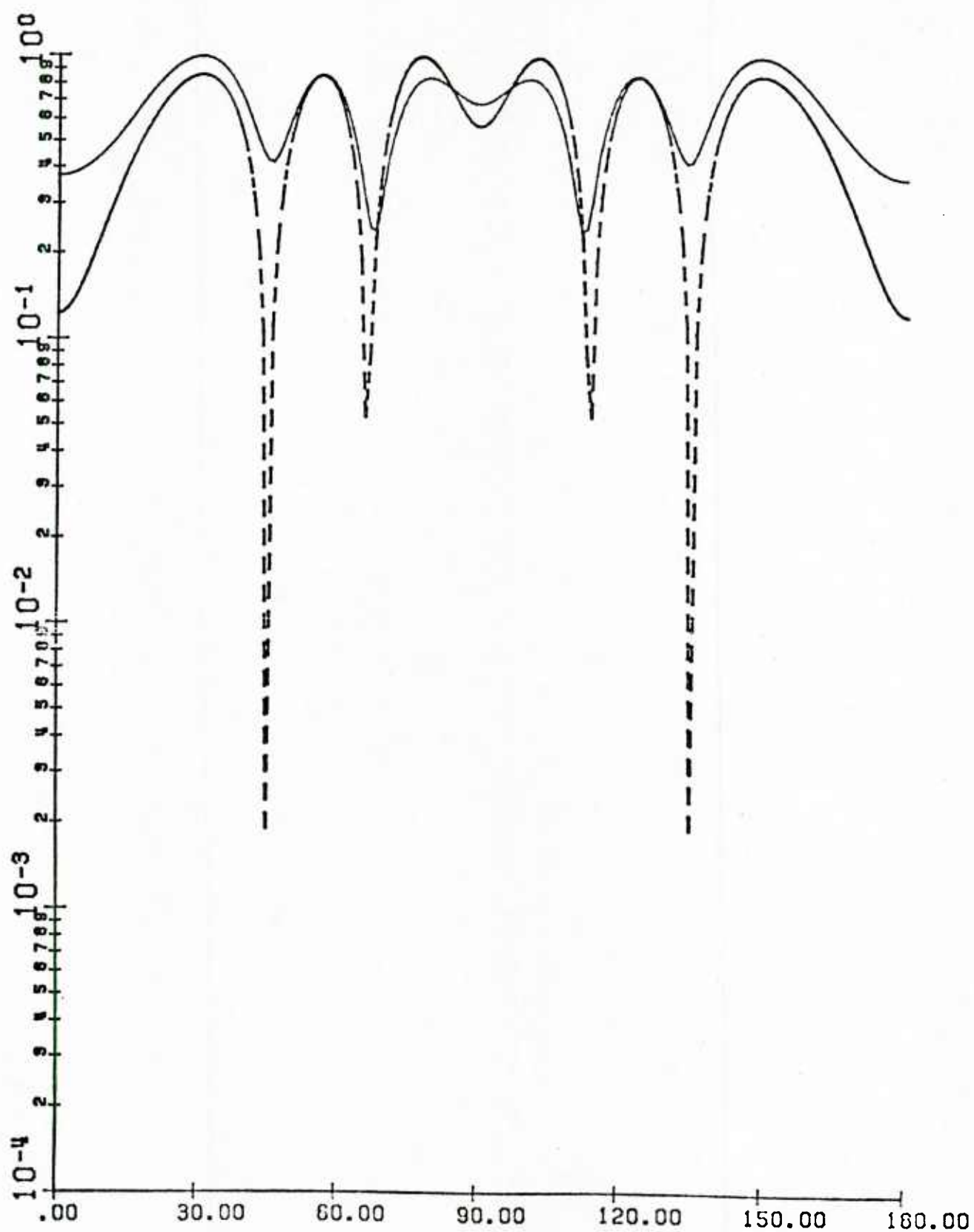


Fig. 17 Array factor in the xz plane of the seventy six antenna planar array with excitation to give a 135 degrees scan in the yz plane

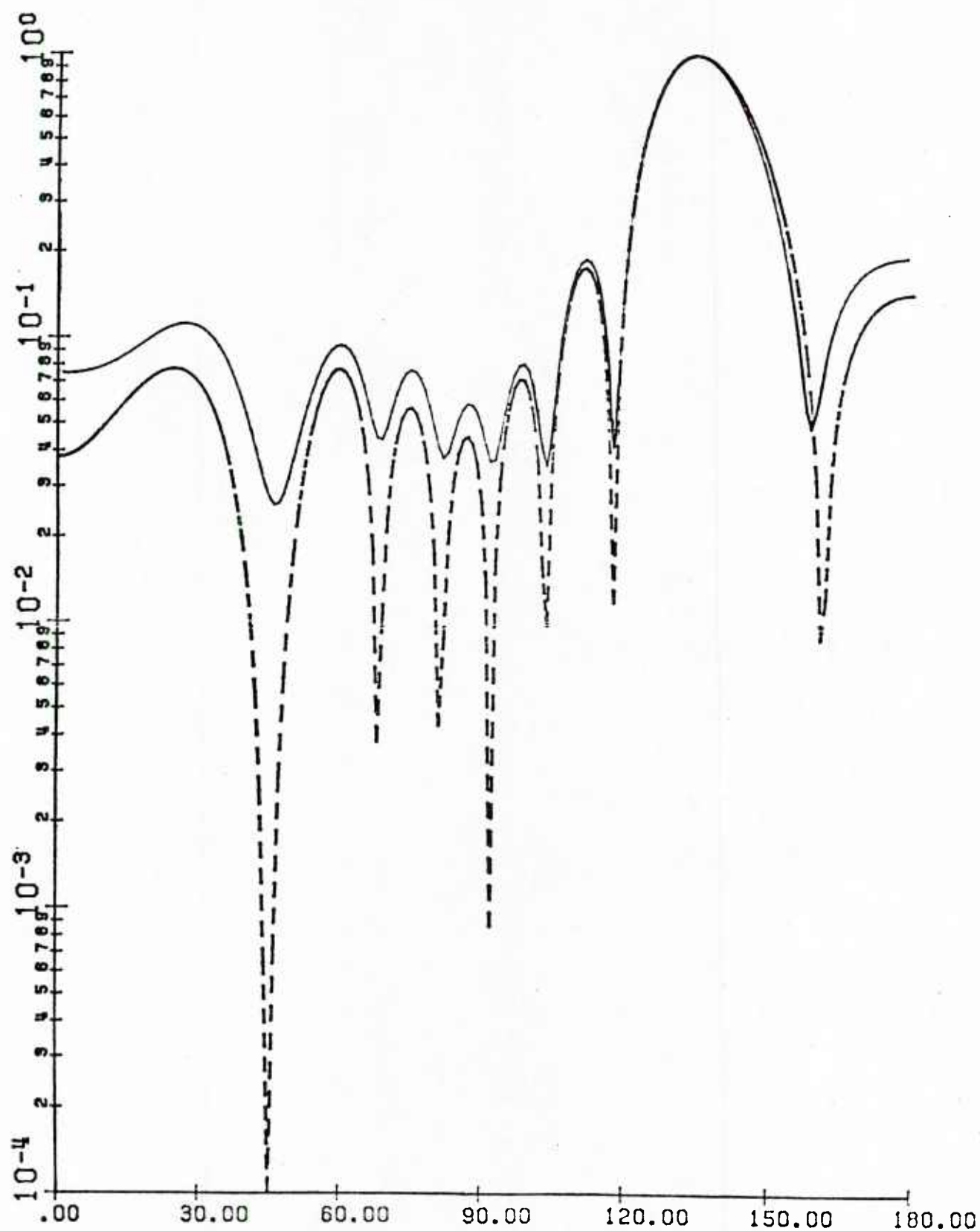


Fig. 18 Array factor in the yz plane of the seventy six antenna planar array with excitation to give a 135 degrees scan in the yz plane

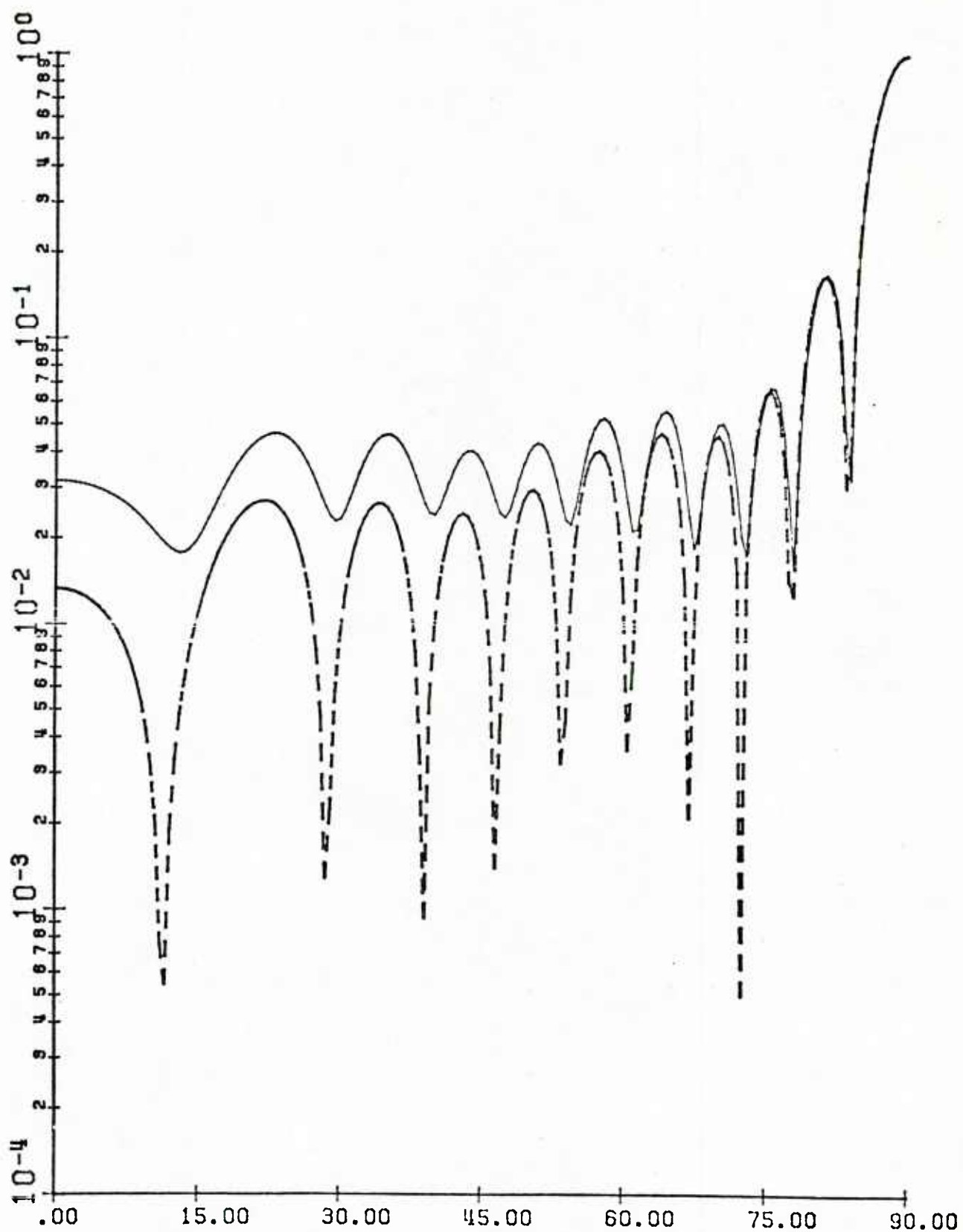


Fig. 19 Array factor in the xz plane of the three hundred and seventy two antenna planar array with uniform excitation

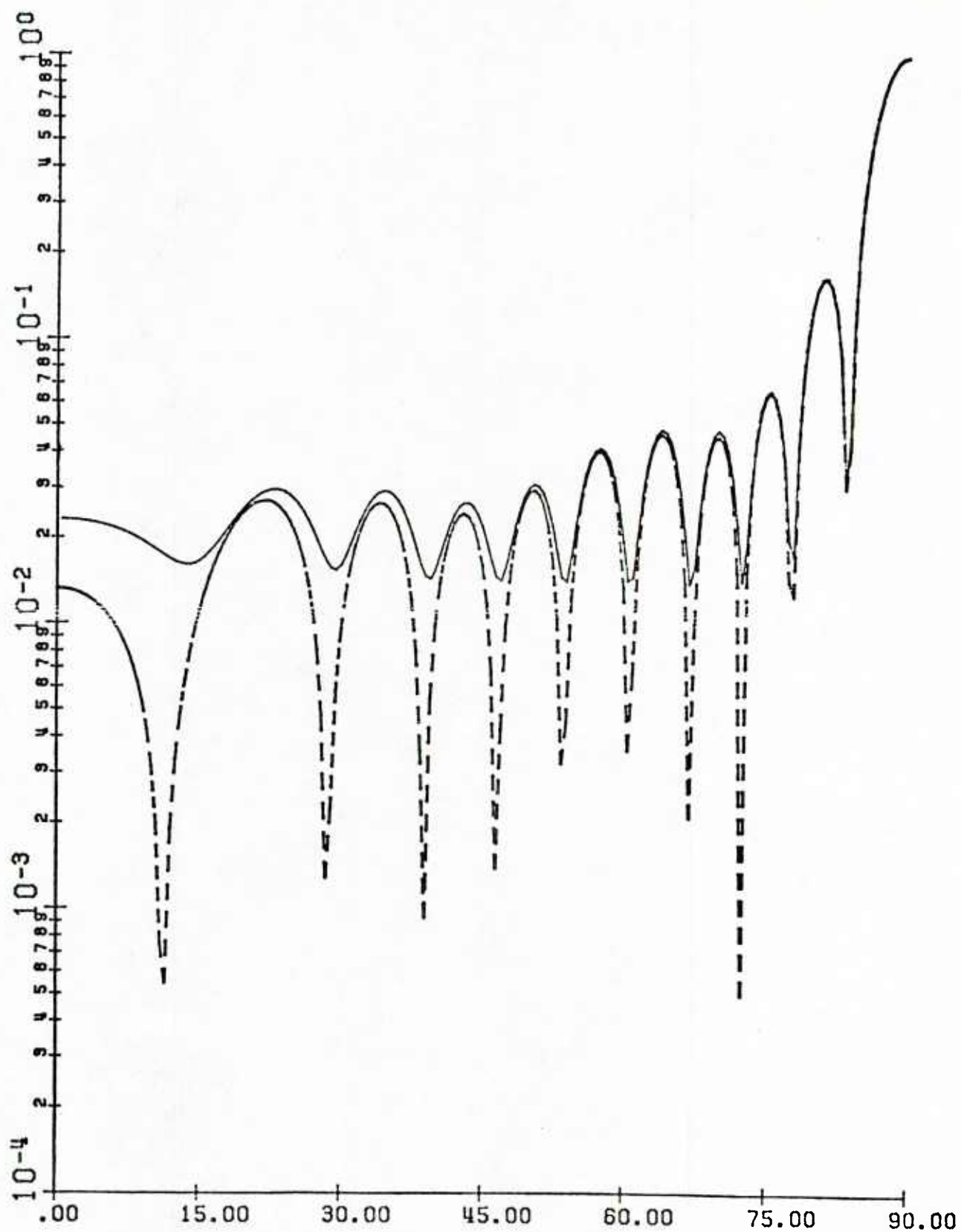


Fig. 20 Array factor in the yz plane of the three hundred and seventy two antenna planar array with uniform excitation

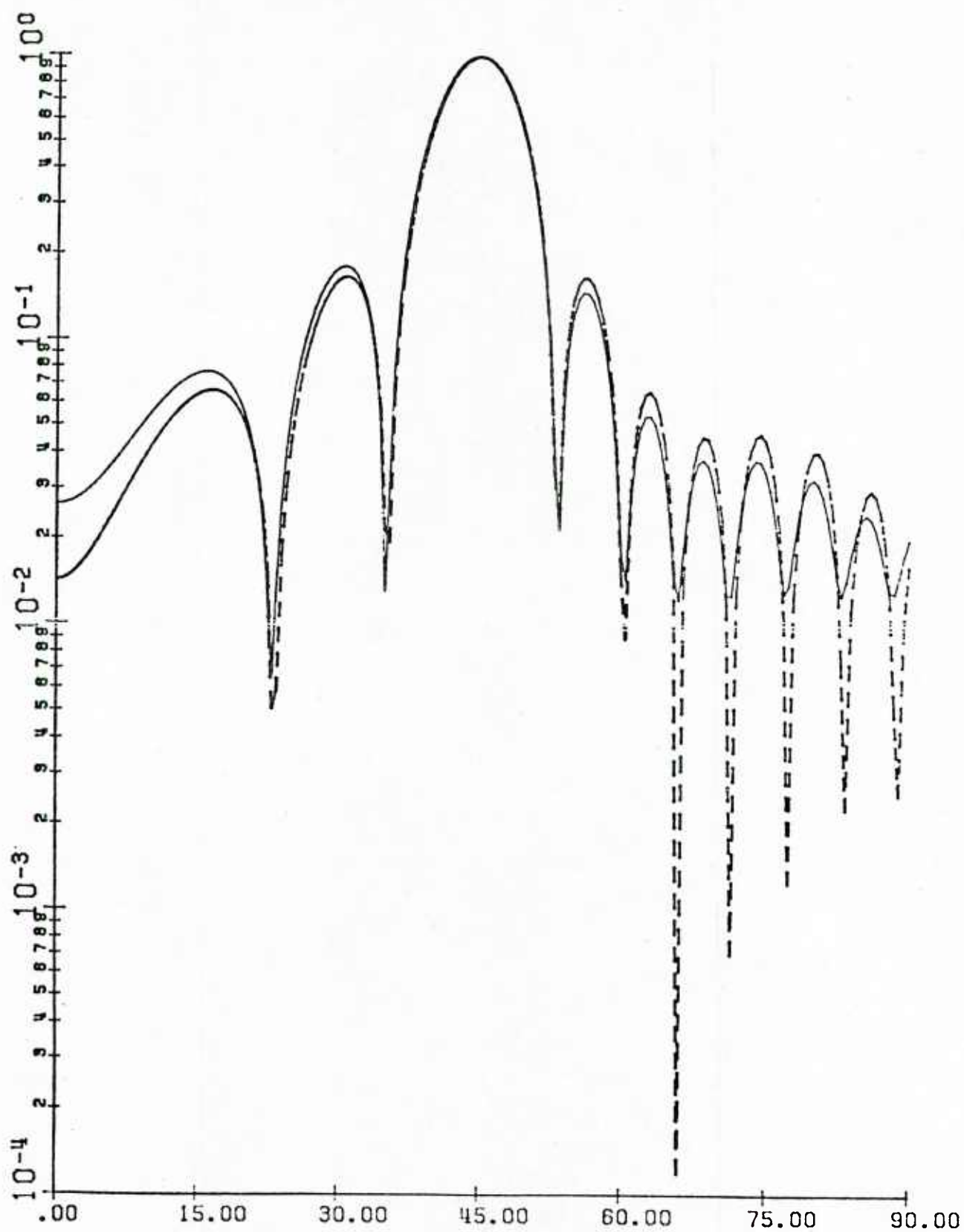


Fig. 21 Array factor in the xz plane of the three hundred and seventy two antenna planar array with excitation to give 45 degrees scan in the xz plane

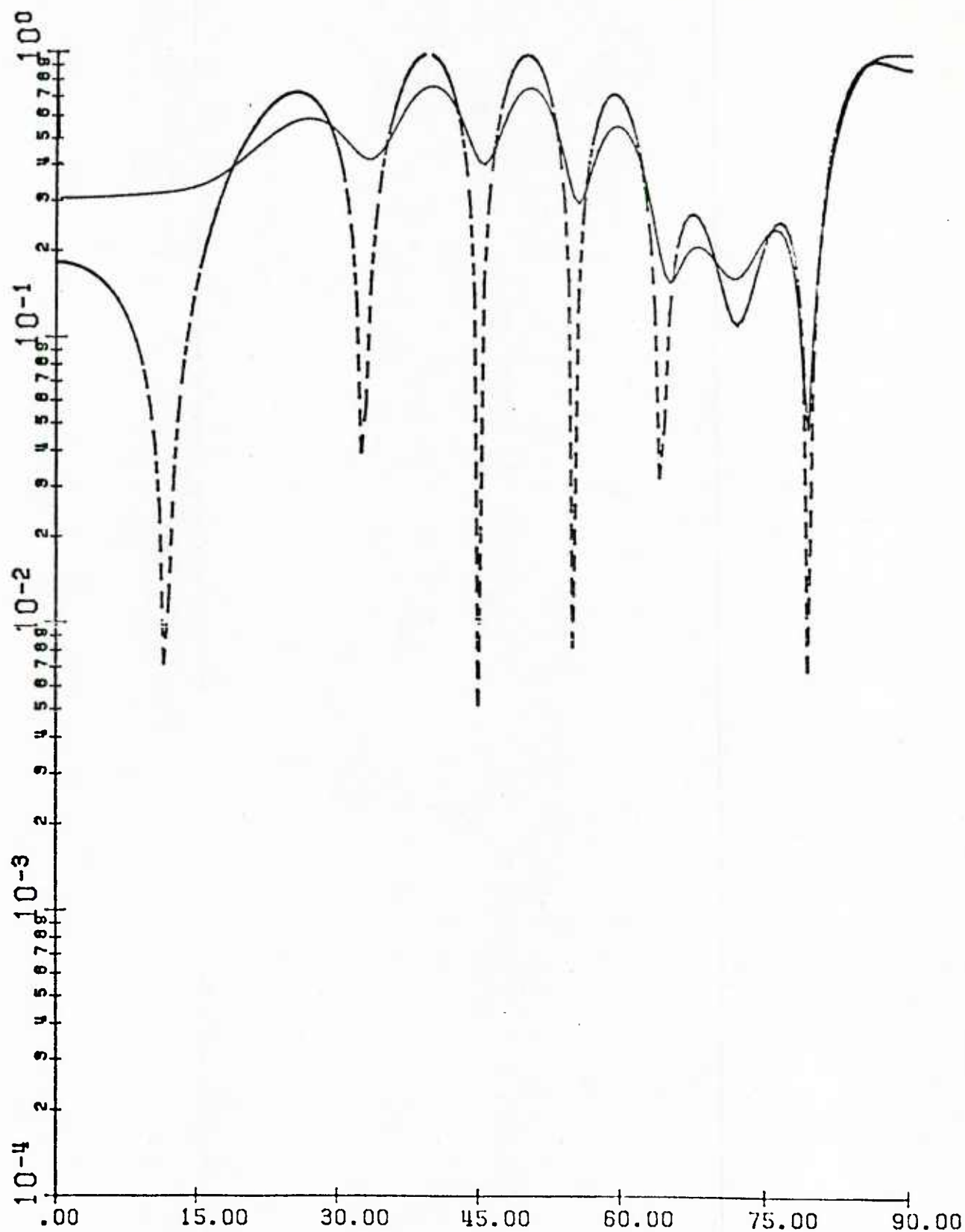


Fig. 22 Array factor in the yz plane of the three hundred and seventy two antenna planar array with excitation to give 45 degrees scan in the xz plane

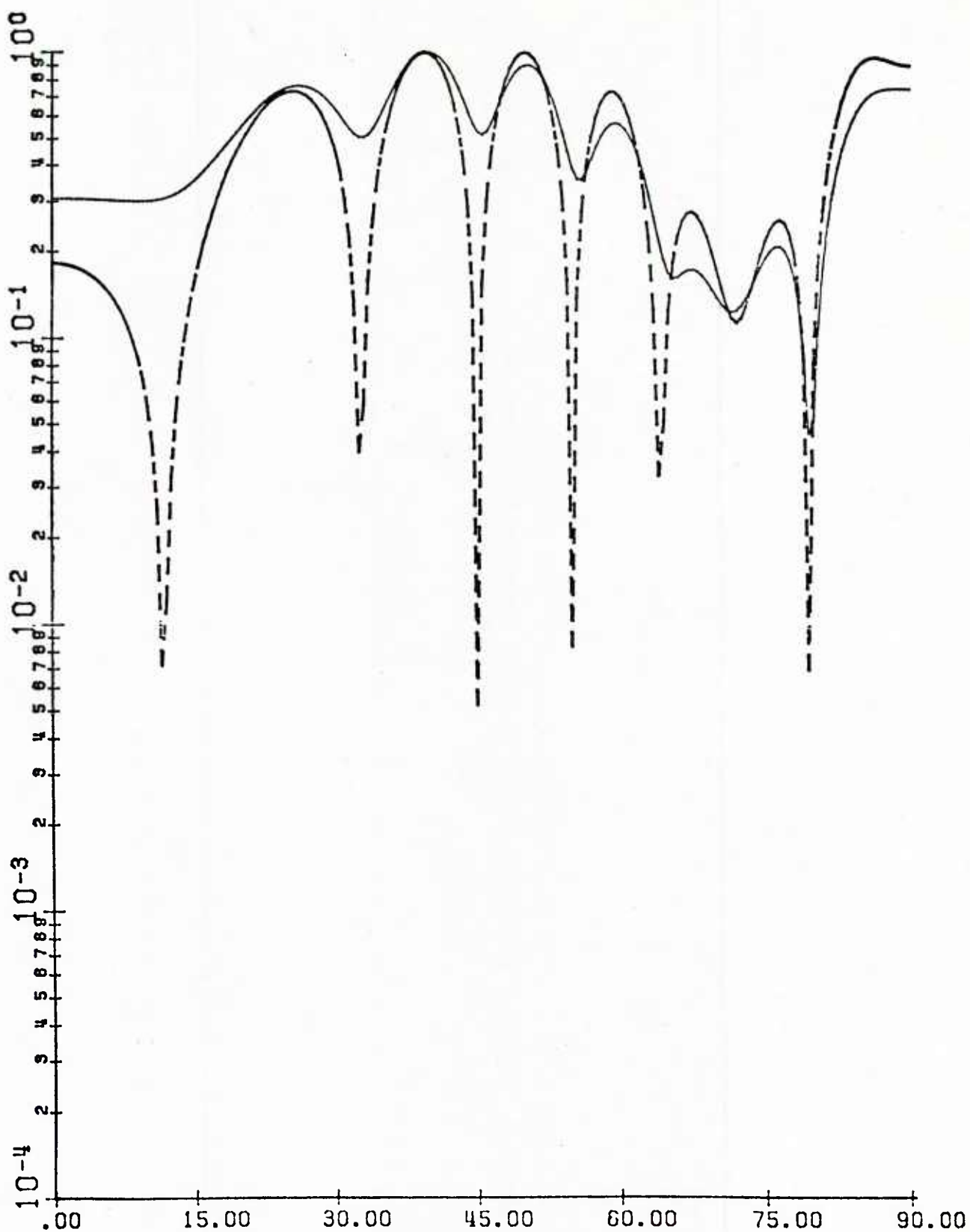


Fig. 23 Array factor in the xz plane of the three hundred and seventy two antenna planar array with excitation to give 135 degrees scan in the yz plane

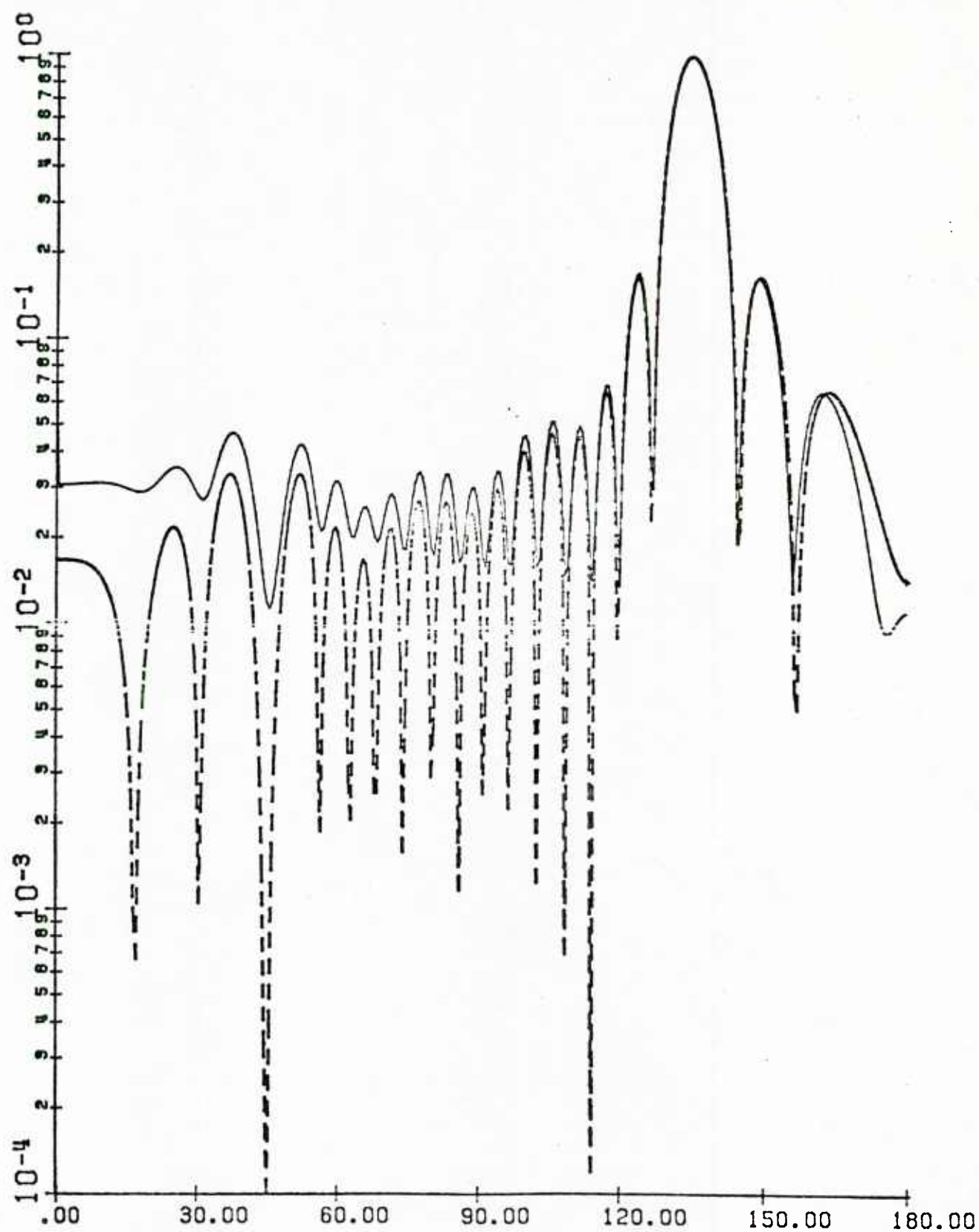


Fig. 24 Array factor in the yz plane of the three hundred and seventy two antenna planar array with excitation to give 135 degrees scan in the yz plane

field pattern of an element with only one expansion function. This is the array factor used in the graphs discussed above.

$$(NAF)_n = \frac{(FP)_n}{(EP)_1} \quad (2)$$

Here $(NAF)_n$ is the normalized array factor for the DCM solution using n expansions per antenna element. $(FP)_n$ is the far field pattern for the DCM solution using n expansion functions per antenna element. $(EP)_1$ is the element pattern or the far field pattern of an element with only one expansion function per element. Since both $(FP)_n$ and $(EP)_1$ goes to zero when the angles are either 0 or 180 degrees, $(NAF)_n$ is indeterminate at those angles.

As we can see from the graphs, for larger arrays, the main beam is not effected by the mutual coupling but the side lobes and the nulls are effected strongly. In addition we can see that the difference between the actual and the ideal array factors are more pronounced for the 45 degrees beam steering in the xz plane. This is expected since the coupling between the antenna elements that are adjacent in the x-direction is stronger than the coupling between the antenna elements that are adjacent in the y-direction. Also, we can see that the array factors computed from the solutions using one expansion function per antenna and the

array factors computed from the solutions using three expansion functions per antenna are surprisingly close. This indicates that at least for the given spacings and antenna sizes, the solutions using only one expansion function per antenna may be good enough for most purposes.

III. CONCLUSIONS

Three more types of problems suggested in the first report on DCM [1] are solved and the number of iterations needed are found to be reasonably small for the antenna array problems. The helix problems required more iterations but the number of iterations needed for the required degree of accuracy is found to be practically independent of the length of the helix.

Three other major types of problems formulated or suggested in [1] still remains to be solved. They are

- (i) scattering from a arbitrarily shaped planer
conducting surfaces or dielectric sheets
- (ii) scattering from a imperfectly conducting or
dielectric bodies
- (iii) planar array antenna backed by a finite sized
ground plane

The major question here is whether or not convergence will be achieved for these types of problems. The condition for convergence is that the absolute value of the largest

eigenvalue of $[Z]$ must be less than 1 as shown in [1]. There are some practical applications where the solutions of these types of problems are useful. Therefore, solving some representative problems are suggested as possible future extension to this report.

APPENDIX A

SPATIAL DOMAIN INTERPRETATION OF THE SPECTRAL ITERATION TECHNIQUES

Both the spectral solution techniques(STD,SIT etc.) [2], [3], [4], [5] and the Discrete Convolution Method (DCM) [1] use the discrete Fourier Transform to solve electromagnetic fields problems. The following spatial domain interpretation of the spectral solution techniques may give a better understanding of the differences between the methods.

The electric field integral equation for a perfectly conducting scatterer or radiator is [2]

$$(\vec{\bar{G}} * \vec{J})_t = -\vec{E}_t^i \quad r \in S \quad (A1)$$

Here $\vec{\bar{G}}$ is the Green's Dyadic, \vec{J} is the surface current density on surface S and \vec{E}_t^i is the tangential component of the impressed field. We can extend the field equation over all space so that (A1) becomes

$$\vec{\bar{G}} * \vec{J} = -\vec{E}^i + \vec{F} \quad (A2)$$

where \vec{F} is the field outside the surface S . The Fourier transformed version of (A2) reads

$$\begin{aligned}\tilde{G}(\Omega) \tilde{J}(\Omega) &= -\tilde{E}_I(\Omega) + \tilde{F}(\Omega) \\ \Omega &= (\omega^x, \omega^y, \omega^z) \text{ in general}\end{aligned}\tag{A3}$$

A formal solution to (A3) is

$$\tilde{J}(\Omega) = \tilde{G}^{-1}(\Omega) \tilde{E}(\Omega)\tag{A4}$$

where $\tilde{E}(\Omega)$ is $-E_I(\Omega) + F(\Omega)$.

The spectral solution techniques essentially consists of iterative techniques that find \vec{F} and thus \vec{J} . However since \vec{F} cannot be found in closed form, the solution is a numerical solution. Therefore (A4) and hence (A3) is solved only for a finite number of specific discrete frequencies. The solution we get is an exact solution of (A3) and hence of (A1) only if \vec{F} is known exactly and if the solution is for all frequencies. In the following discussion we will set aside the question of inaccuracies due to inexact knowledge of \vec{F} because it is a question of convergence. Here we are discussing spatial domain interpretation rather than the convergence of the solution. Therefore

(a) since (A3) is satisfied only for discrete frequencies, we are sampling in frequency domain and we are solving the following set of equations

$$\tilde{G}(\Omega) \tilde{J}(\Omega) \delta(\Omega - \Omega_n) = \tilde{E}(\Omega) \delta(\Omega - \Omega_n) \quad (A5)$$

$$\Omega_n = (\omega_a^x, \omega_b^y, \omega_c^z)$$

$$\omega_a^x = \omega_0^x, \omega_1^x, \dots$$

$$\omega_b^y = \omega_0^y, \omega_1^y, \dots$$

$$\omega_c^z = \omega_0^z, \omega_1^z, \dots$$

instead of (A3).

(b) Since we cannot solve for infinite number of discrete frequencies, we solve for only a finite number. This amounts to truncating in frequency domain. If we denote the truncating function by $\tilde{H}(\Omega)$, we are solving the following set of equations

$$\tilde{H}(\Omega) \tilde{G}(\Omega) \tilde{J}(\Omega) \delta(\Omega - \Omega_n) = \tilde{H}(\Omega) \tilde{E}(\Omega) \delta(\Omega - \Omega_n) \quad (A6)$$

instead of either (A3) or (A5).

Before we give the spatial domain interpretation of (A6), we will first examine the errors due to above approximations from a frequency domain perspective. It is well known [6] that sampling in the spatial domain will lead to aliasing in frequency domain if the function sampled is not limited in frequency. By reason of symmetry, we can easily show that sampling in frequency domain will lead to aliasing in the spatial domain if the function sampled is not limited in space. In solving (A3), we used sampling on both sides of the equation. The surface current \vec{J} is indeed limited in space (being confined to conductor surface S) but

the field \vec{E} is not. Therefore, there will be aliasing irrespective of the sampling rate. However, since the field \vec{E} decays outside the surface, aliasing effect can be reduced by a high enough sampling rate so that the overlap occurs in the region where \vec{E} has already decayed to a small value. Truncation of the frequency range by $\tilde{H}(\Omega)$ (the windowing function) on both sides of the equation leads to Gibb's phenomenon. This windowing error can be reduced by using different windowing strategies discussed in [6]. A good example is the Hamming windowing function.

We will now look at the above errors from a spatial domain perspective. In spectral solution techniques, solution to (A6) is found by using the Discrete Fourier Transform. This is possible if Ω_n 's are chosen at equally spaced intervals. It can be proved [6] that (A6) in this case is equivalent to the Discrete Convolution equation

$$\begin{aligned} \bar{g}(K) * \vec{J}(K) &= \vec{E}(K) \quad K=(k_x, k_y, k_z) \\ k_x &= 0, 1, \dots, L \\ k_y &= 0, 1, \dots, M \\ k_z &= 0, 1, \dots, N \end{aligned} \tag{A7}$$

Up to this point, we have been discussing the general multi-dimensional case. From here on however, we will confine our discussion to the one dimensional case. Discussion for two or three dimensional problems would be similar. (We will use lower case indices instead of higher

case, to indicate one dimensionality.) Since \vec{E}_I is known exactly and \vec{F} , once we achieve convergence, is also known exactly, $\vec{E}(k)$ represents the sampled value at k^{th} point of the exact field \vec{E} [2], [5]. Also, \vec{J} is the required solution and hence $\vec{J}(k)$ represents the sampled value at k^{th} point of the current \vec{J} . But $\bar{\vec{g}}(k)$ is not the sampled value of $\bar{\vec{G}}$. Instead, $\bar{\vec{g}}(k)$ is the inverse Discrete Fourier Transform (based on $N+1$ points) of $\bar{\vec{G}}(\omega)$.

$$\bar{\vec{g}}(k) = \sum_{n=-N/2}^{N/2} \bar{\vec{G}}(n) e^{j(\frac{2\pi}{N+1})kn} \quad (\text{A8})$$

Therefore $\bar{\vec{g}}(k)$ is the inverse Fourier Transform of the sampled and truncated (or in general windowed) $\bar{\vec{G}}(\omega)$.

It is well known [6] that sampling in spatial domain will lead to a function in frequency domain which consists of periodic repetitions of the original frequency domain function. By reason of symmetry between the domains, we can easily show that sampling in frequency domain will lead to a function in spatial domain which consists of periodic repetitions of the original spatial domain function. Therefore, sampling of $\bar{\vec{G}}(\omega)$ will cause the inverse of the sampled function $\bar{\vec{g}}_s$ to consist of periodic repetitions of the original spatial domain function $\bar{\vec{G}}$. This will cause overlaps as shown in Fig. 25 for the one dimensional case. But G is the Green's Dyadic, the field due to a unit impulse

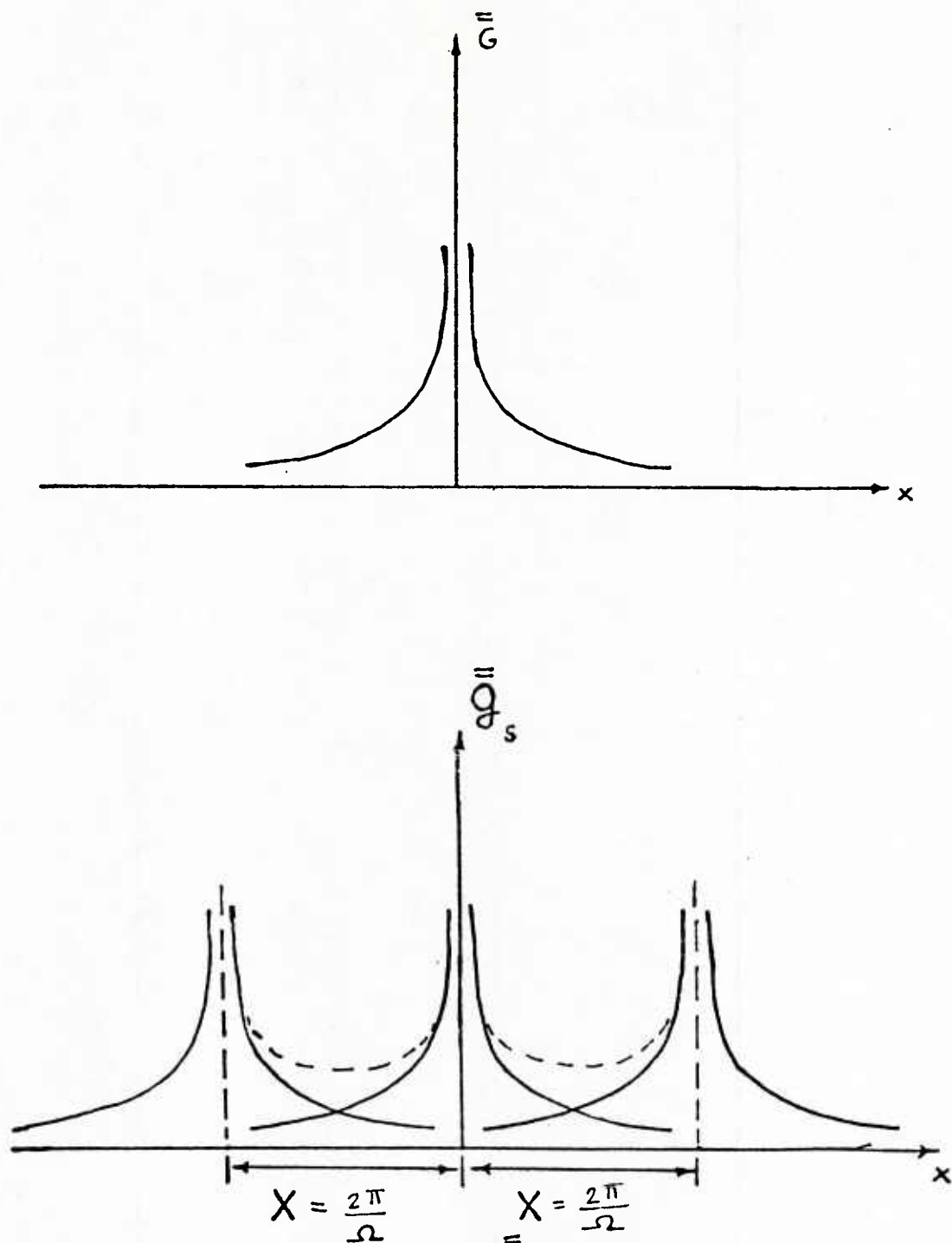


Fig. 25 Relationship between \bar{q}_s and \bar{G} for the one dimensional case

current at the origin. Therefore $\bar{\bar{g}}_s$ is the overlapped approximate field of a unit impulse current at the origin. Using higher sampling rates will cause the rate of repetition to decrease, i.e., increase X . Since we are interested in solving (A7) for only a finite region in space, error due to overlap will be insignificant if the sampling rate is high enough.

However, in addition to sampling, we also truncate the frequency function. This truncation will, in general, be done by using a windowing function $\tilde{H}(\omega)$. Therefore $\bar{\bar{G}}(k)$ can be thought of as the inverse Fourier Transform of the sampled version of the function $\bar{\bar{G}}(\omega)\tilde{H}(\omega)$. $\bar{\bar{G}}(\omega)$ is the Fourier Transform of the Green's Dyadic $\bar{\bar{G}}$, the field due to a unit current impulse at the origin. Therefore $\bar{\bar{G}}(\omega)\tilde{H}(\omega)$ is the Fourier Transform of the field due to a current distribution \vec{H} , where \vec{H} is the inverse Fourier Transform of the windowing function $\tilde{H}(\omega)$. Therefore $\bar{\bar{g}}(k)$ is the sampled value at k^{th} point of the field due to the current distribution \vec{H} . Therefore, the result of the convolution product $\bar{\bar{g}}*\vec{J}$ is the (overlapped) approximate field due to the sampled values of the current \vec{J} multiplied by the current distribution \vec{H} . Since a convolution product produces a finite set of values, each one corresponding to a point in space, each value can be interpreted as the approximate field at the corresponding point in space due to the current \vec{J} , expanded in terms of the basis or expansion function \vec{H} . Since solving (6) amounts to matching the left

hand side and the right hand side of (A7) [6], the solution of (A6) can now be seen as point matching the field E with the approximate field due to \vec{J} expanded in terms of the basis or expansion function \vec{H} . If the error due to overlap is not significant, then the solution of (A6) is equivalent to point matching the known field E with the field due to \vec{J} expanded in terms of the basis or expansion function \vec{H} . Therefore, the spectral iteration method is equivalent to the Method of Moments [7], if we use the inverse Fourier Transform of the windowing function $\tilde{H}(\omega)$ as expansion function and point match. The only difference is that with the Method of Moments, Gaussian elimination is normally used instead of iterative techniques.

The Discrete Convolution Method [1] on the other hand, is the iterative solution of the matrix equation formulated by the Method of Moments. Therefore the difference between the Discrete Convolution Method and the spectral iteration techniques is that with the former method, the expansion function is chosen explicitly whereas with the latter, the expansion function is chosen implicitly; i.e. the inverse Fourier Transform of the windowing function.

APPENDIX B

COMPUTER PROGRAMS

The computer programs given in this section are written to solve the three additional types of problems. Although they are not optimized in any sense of the word, they are written to avoid obvious wastes of computing time and memory space and is fairly efficient.

I. MAIN PROGRAM SEGMENT AND SUBROUTINES FOR THE HELIX PROBLEM

Both the main program segment and the subroutine CALZ are from [8]. The slight modifications in the main program are self-explaining so no attempt will be made here to give a description. The subroutine HELIX on the other hand, is written to compute the co-ordinates of the helix points from the given radius, pitch, number of turns, and the number of points for which the co-ordinates must be found.

```

400 C
500 C      THIS IS THE MAIN PROGRAM
600 C
700      COMPLEX Z(3600),U(400),C(400),E(3),EI(2),UV(4)
800      COMPLEX U1,ZL(30),ZIN,YIN,V,ZC
900      COMPLEX TOE(1024),EX(400),CUR(1024),GUESS
000      COMMON /G/GUESS,CUR
100      COMMON XX(800),XY(800),XZ(800),TX(800),TY(800),TZ(800),AL(800)
200      COMMON T(1600),TP(1600),BK,RAD2(21),L(11),LR(21)/COA/C
300      DIMENSION PX(900),PY(900),PZ(900),LL(11),RAD(21),IFP(60)
400      DIMENSION LP(30)
500      PI=3.141592654
550      OPEN(UNIT=21,FILE='ARDATA.DAT')
600 1      READ(1,101) NW,NP,NR,BK,IFLAG
700      WRITE(3,102) NW,NP,NR,BK
800      LL(NW+1)=801
900      LR(NR+1)=801
000      IF(IFLAG.NE.0) GO TO 300
100      READ(1,130) (PX(I),I=1,NP)
200      READ(1,130) (PY(I),I=1,NP)
300      READ(1,130) (PZ(I),I=1,NP)
400      GO TO 301
500      300      CONTINUE
510      IF(IFLAG.NE.1) GO TO 310
520      CALL HELIX(PX,PY,PZ,NP)
530      GO TO 301
540      310      CONTINUE
550      READ(1,130) WIREL,SEGL
600      DO 501 I=1,NP
700      PX(I)=WIREL
800      WIREL=WIREL+SEGL
900      PY(I)=0.0
000      PZ(I)=0.0
100      501      CONTINUE
200      301      CONTINUE
300      WRITE(3,104) (PX(I),I=1,NP)
400      WRITE(3,105) (PY(I),I=1,NP)
500      WRITE(3,106) (PZ(I),I=1,NP)
600      READ(1,107) (LL(I),I=1,NW)
700      WRITE(3,108) (LL(I),I=1,NW)
800      READ(1,107) (LR(I),I=1,NR)
900      WRITE(3,103) (LR(I),I=1,NR)
000      READ(1,109) (RAD(I),I=1,NR)
100      WRITE(3,110) (RAD(I),I=1,NR)
200      101      FORMAT(3I3,E14.7)
300      102      FORMAT('0NW NP NR      BK'/3I3,E14.7)
400      103      FORMAT(10F0.0)
500      104      FORMAT('0PX'/(1X,8F8.4))
600      105      FORMAT('0PY'/(1X,8F8.4))
700      106      FORMAT('0PZ'/(1X,8F8.4))
800      107      FORMAT(20I3)
900      108      FORMAT('0LL'/(1X,10I4))
000      109      FORMAT(5E14.7)
100      103      FORMAT('0LR'/(1X,10I4))
200      110      FORMAT('0RAD'/(1X,8E0.0))
300      DO 46 I=1,NR
400      RAD2(I)=RAD(I)*RAD(I)
500      46      CONTINUE
600      J1=1
700      J2=2
800

```



```

0900      N1=0
1000      DO 2 J=1,NP
1100      IF (L(J1)-J) 3,4,3
1200      4 J2=J2-1
1300      L(J1)=J2
1400      J1=J1+1
1500      GO TO 2
1600      3 N1=N1+1
1700      J3=J-1
1800      IF ((N1/2*2-N1).EQ.0) J2=J2+1
1900      XX(N1)=.5*(PX(J)+PX(J3))
2000      XY(N1)=.5*(PY(J)+PY(J3))
2100      XZ(N1)=.5*(PZ(J)+PZ(J3))
2200      S1=PX(J)-PX(J3)
2300      S2=PY(J)-PY(J3)
2400      S3=PZ(J)-PZ(J3)
2500      S4=SQRT(S1*S1+S2*S2+S3*S3)
2600      TX(N1)=S1/S4
2700      TY(N1)=S2/S4
2800      TZ(N1)=S3/S4
2900      AL(N1)=S4
3000      2 CONTINUE
3100      L(J1)=J2
3200      N=J2-2
3300      CALL CALZ(N,N1,Z)
3400      WRITE(21,113) (Z(I),I=1,N)
3500      113 FORMAT(5E14.7)
3600      DO 45 I=1,N
3700      U(I)=0.
3800      45 CONTINUE
3900      READ(1,107) NSET
4000      WRITE(3,127) NSET
4100      IF(NSET) 33,33,32
4200      127 FORMAT('ONSET'/I4)
4300      32 DO 26 III=1,NSET
4400      U1=(0.,1.)
4500      READ(1,126) THE,PHI,EI(1),EI(2)
4600      126 FORMAT(8E0.0)
4700      GUESS=GUESS*RAD(1)/188.365
4800      WRITE(3,125) THE,PHI,EI(1),EI(2)
4900      125 FORMAT('0*****'/ '0THETA PHI EI(1)',
5000      ' ' EI(2)'/2F6.0,4E11.4)
5100      1 A1=CABS(EI(1))**2+CABS(EI(2))**2
5200      A2=BK*BK
5300      TH=THE*.0174533
5400      PH=PHI*.0174533
5500      CT=COS(TH)
5600      ST=SIN(TH)
5700      CP=COS(PH)
5800      SP=SIN(PH)
5900      S1=CT*CP
6000      S2=CT*SP
6100      BK1=BK*ST*CP
6200      BK2=BK*ST*SP
6300      BK3=BK*CT
6400      J1=1
6500      J2=-2
6600      DO 27 J=1,N
6700      IF (L(J1)-J) 29,28,29
6800      28 J2=J2+2

```



```

7600      J1=J1+1
7700      KK=1
7800      GO TO 30
7900  29    UV(1)=UV(3)
8000      UV(2)=UV(4)
8100      KK=3
8200  30    DO 31 M=KK,4
8300      J3=J2+M
8400      XDT=TX(J3)*S1+TY(J3)*S2-TZ(J3)*ST
8500      XDP=-TX(J3)*SP+TY(J3)*CP
8600      BKR=XX(J3)*BK1+XY(J3)*BK2+XZ(J3)*BK3
8700      UV(M)=(XDT*EI(1)+XDP*EI(2))*(COS(BKR)+U1*SIN(BKR))
8800  31    CONTINUE
8900      J3=(J-1)*4
9000      J4=J3+1
9100      J5=J4+1
9200      J6=J5+1
9300      J7=J6+1
9400      U(J)=T(J4)*UV(1)+T(J5)*UV(2)+T(J6)*UV(3)+T(J7)*UV(4)
9500      J2=J2+2
9600  27    CONTINUE
9700      WRITE(21,113)(U(I),I=1,N)
9800  26    CONTINUE
9900  33    CONTINUE
1000      WRITE(3,112)
1010  112   FORMAT(1H,'TYPE 1 TO STOP, ELSE 0 AND RETURN')
1020      READ(1,101) IFLAG
1030      IF(IFLAG.EQ.0) GO TO 1
1040      CLOSE(UNIT=21)
1050      STOP
1060      END
1070
1080  C
1090  C      THIS IS SUBROUTINE #1
1100  C
1110      SUBROUTINE CALZ(N,N1,Z)
1120      COMPLEX Z(3600),PSI(3200),U,U1,U2,U3,U4,U5,U6
1130      COMMON XX(800),XY(800),XZ(800),TX(800),TY(800),TZ(800),AL(800)
1140      COMMON T(1600),TP(1600),BK,RAD2(21),L(11),LR(21)
1150      DIMENSION DC(3200)
1160      U=(0.,1.)
1170      PI=3.141593
1180      ETA=376.7307
1190      C1=.125/PI
1200      C2=.25/PI
1210      J1=1
1220      J2=-2
1230      DO 1 J=1,N
1240      IF(L(J1)-J) 3,4,3
1250  4      J2=J2+2
1260      J1=J1+1
1270  3      J3=(J-1)*4
1280      J4=J3+1
1290      J5=J4+1
1300      J6=J5+1
1310      J7=J6+1
1320      K4=J2+1
1330      K5=K4+1
1340      K6=K5+1
1350      K7=K6+1
1360      S1=AL(K4)+AL(K5)

```

```

4200      S2=AL (K6) +AL (K7)
4300      T (J4)=AL (K4) *.5*AL (K4) /S1
4400      T (J5)=AL (K5) * (AL (K4) +.5*AL (K5)) /S1
4500      T (J6)=AL (K6) * (AL (K7) +.5*AL (K6)) /S2
4600      T (J7)=AL (K7) *.5*AL (K7) /S2
4700      TP (J4)=AL (K4) /S1
4800      TP (J5)=AL (K5) /S1
4900      TP (J6)=-AL (K6) /S2
5000      TP (J7)=-AL (K7) /S2
5100      J2=J2+2
5200      1      CONTINUE
5300      U3=U*BK*ETA
5400      U4=-U/BK*ETA
5500      BK2=BK*BK/2.
5600      BK3=BK2*BK/3.
5700      N9=0
5800      N2=1
5900      N0=1
6000      N3=-2
6100      DO 10 NS=1,N
6200      IF (L (N2) -NS) 12,11,12
6300      11      KK=1
6400      N3=N3+2
6500      N2=N2+1
6600      GO TO 13
6700      12      KK=3
6800      DO 14 NF=1,N1
6900      N4=NF+N1
7000      N5=N4+N1
7100      N6=N5+N1
7200      DC (NF)=DC (N5)
7300      DC (N4)=DC (N6)
7400      PSI (NF)=PSI (N5)
7500      PSI (N4)=PSI (N6)
7600      14      CONTINUE
7700      13      CONTINUE
7800      DO 15 K=KK,4
7900      N7=N3+K
8000      K1=(K-1)*N1
8100      N0=1
8200      DO 16 NF=1,N1
8300      IF (NF-LR (N0)) 5,6,5
8400      6      AA=RAD2 (N0)
8500      N0=N0+1
8600      5      CONTINUE
8700      N8=NF+K1
8800      S1=XX (N7) -XX (NF)
8900      S2=XY (N7) -XY (NF)
9000      S3=XZ (N7) -XZ (NF)
9100      R2=S1*S1+S2*S2+S3*S3+AA
9200      R=SQRT (R2)
9300      RT=ABS (S1*TX (N7) +S2*TY (N7) +S3*TZ (N7) )
9400      RT2=RT*RT
9500      RH=(R2-RT2)
9600      ALP=.5*AL (N7)
9700      AR=ALP/R
9800      S1=BK*R
9900      U2=COS (S1) -U*SIN (S1)
0000      IF (AR-.1) 22,22,21
0100      21      U2=U2*C1/ALP

```

```

00200      S1=RT-ALP
00300      S2=RT+ALP
00400      S3=SQRT(S1*S1+RH)
00500      S4=SQRT(S2*S2+RH)
00600      IF(S1) 18,18,19
00700 18      AI1=ALOG((S2+S4)*(-S1+S3)/RH)
00800      GO TO 20
00900 19      AI1=ALOG((S2+S4)/(S1+S3))
01000 20      AI2=AL(N7)
01100      AI3=(S2*S4-S1*S3+RH*AI1)/2.
01200      AI4=AI2*(RH+ALP*ALP/3.+RT2)
01300      S3=AI1*R
01400      S1=AI1-BK2*(AI3-R*(2.*AI2-S3))
01500      S2=-BK*(AI2-S3)+BK3*(AI4-3.*AI3*R+R2*(3.*AI2-S3))
01600      GO TO 28
01700 22      U2=U2*C2/R
01800      BA=BK*ALP
01900      BA2=BA*BA
02000      AR2=AR*AR
02100      AR3=AR2*AR
02200      ZR=RT/R
02300      ZR2=ZR*ZR
02400      ZR3=ZR2*ZR
02500      ZR4=ZR3*ZR
02600      H1=(3.-30.*ZR2+35.*ZR4)*AR3/40.
02700      A1=AR*(-1.+3.*ZR2)/6.+(3.-30.*ZR2+35.*ZR4)*AR3/40.
02800      A0=1.+AR*A1
02900      A2=-ZR2/6.-AR2*(1.-12.*ZR2+15.*ZR4)/40.
03000      A3=AR*(3.*ZR2-5.*ZR4)/60.
03100      A4=ZR4/120.
03200      S1=A0+BA2*(A2+BA2*A4)
03300      S2=BA*(A1+BA2*A3)
03400 28      PSI(N8)=U2*(S1+U*S2)
03500      DC(N8)=TX(NF)*TX(N7)+TY(NF)*TY(N7)+TZ(NF)*TZ(N7)
03600 16      CONTINUE
03700 15      CONTINUE
03800      N3=N3+2
03900      J3=(NS-1)*4
04000      J7=-2
04100      J9=1
04200      DO 25 NF=1,N
04300      J1=(NF-1)*4
04400      IF(L(J9)-NF) 26,27,26
04500 27      J9=J9+1
04600      J7=J7+2
04700 26      N9=N9+1
04800      U5=0.
04900      U6=0.
05000      J5=0
05100      DO 23 JS=1,4
05200      J4=J3+JS
05300      J8=J5+J7
05400      DO 24 JF=1,4
05500      J6=J8+JF
05600      J2=J1+JF
05700      U5=T(J2)*T(J4)*DC(J6)*PSI(J6)+U5
05800      U6=TP(J2)*TP(J4)*PSI(J6)+U6
05900 24      CONTINUE
06000      J5=J5+N1
06100 23      CONTINUE

```

```

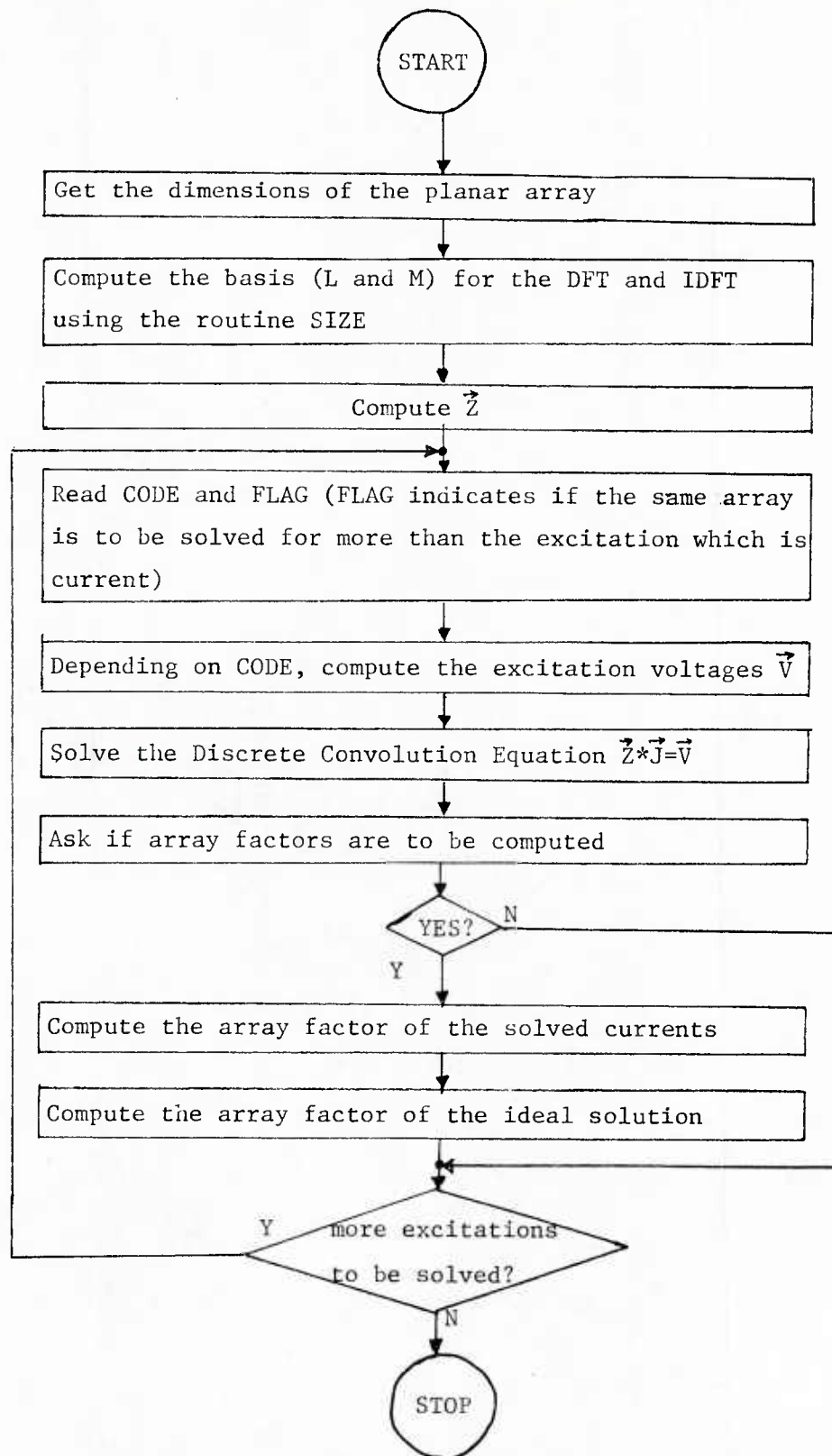
06200      Z(N9)=U5*U3+U6*U4
06300      J7=J7+2
06400      IF(N9.GE.N) RETURN
06500 25     CONTINUE
06600 10     CONTINUE
06700      RETURN
06800      END
06900      SUBROUTINE HELIX(X,Y,Z,NP)
07000      REAL X(1),Y(1),Z(1),RAD,PITCH,PHI,DPHI
07100      INTEGER TURNS
07200      READ(1,100) RAD,PITCH,TURNS
07300      WRITE(3,110) RAD,PITCH,TURNS
07400 100     FORMAT(2F0.0,I0)
07500 110     FORMAT(1H,'RADIUS=',E14.7/
07600      $      1H,'PITCH =',E14.7/
07700      $      1H,'NO OF TURNS=',I7)
07800      PITCH=PITCH/6.2831853
07900      PHI=0.0
08000      DPHI=6.2831853*TURNS/FLOAT(NP-1)
08100      DO 10 I=1,NP
08200      X(I)=RAD*COS(PHI)
08300      Y(I)=RAD*SIN(PHI)
08400      Z(I)=PITCH*PHI
08500      PHI=PHI+DPHI
08600 10     CONTINUE
08700      RETURN
08800      END

```

II. MAIN PROGRAM SEGMENT AND SUBROUTINES FOR THE PLANAR ARRAY PROBLEMS (SINGLE EXPANSION FUNCTION PER ANTENNA ELEMENT)

Subroutine SHAPE zerorize the corners i.e. it zerorize the phantom antenna element currents. Subroutine PATTRN computes the array factor for the given set of array currents. The main program segment solves the planar array problems using the solution routine SOLVE and other routines not listed in this section. They are listed in section III since these routines are common to both single and multiple expansion solution programs.

The flow chart for the main program segment is as follows.



```

00100 C      PROGRAM TO COMPUTE MUTUAL, IMAGE AND TOTAL IMPEDANCES
00200      LOGICAL FXCITE
00300      INTEGER CODE, FLAG, POSPTR, FLAG2
00400      COMPLEX MUTUAL, IMAGE, ZMNG, V(1849), Z(16384), CZERO, Y(1849)
00500 C      READ IN NROWS - NUMBER OF ROWS OF ANTENNA ELEMENTS
00600 C      NCOLS - NUMBER OF COLUMNS
00700 C      DX - SEPERATION BETWEEN ELEMENTS IN X DIRECTION
00800 C      DY - DISTANCE OF ELEMENTS FROM GROUND PLANE
00900 C      DZ - SEPERATION BETWEEN ELEMENTS IN Z DIRECTION
01000      FXCITE=.TRUE.
01100      TWOPI=6.2831853
01200      CZERO=(0.0,0.0)
01300      OPEN(UNIT=21, FILE='PATTRN.DAT')
01400      READ(1,100) NSIDE,CCODE,DX,DY,DZ,WLNGTH,RAD
01500      NT=NSIDE*3-2
01600      NROWS=NT
01700      NCOLS=NT
01800      CALL SIZE(NCOLS,NROWS,LM,MM,L,M,N)
01900      DO 1 I=1,N
02000      Z(I)=CZERO
02100 1    CONTINUE
02200      DX=DX*TWOPI
02300      DY=DY*TWOPI
02400      DZ=DZ*TWOPI
02500      WLNGTH=WLNGTH*TWOPI
02600      RAD=RAD*TWOPI
02700      NELEM=NROWS*NCOLS
02800      ZFLEM=0.0
02900      DY2SQ=4.0*DY*DY
03000      RAD2=RAD*RAD
03100      WLBY2=WLNGTH/2.0
03200      POSPTR=1
03300      DO 20 I=1,2*NT-1
03400      XCUR=(I-1)*DX
03500      ZCUR=(6*NSIDE-5-I)*DZ
03600      DO 10 J=1,2*NT-1
03700      RR=XCUR*XCUR+DY2SQ
03800      IMAGE=ZMNG(ZELEM,ZCUR,WLBY2,WLBY2,RR)
03900      RR=RR-DY2SQ
04000      IF(RR.EQ.0.0) RR=RR+RAD2
04100      MUTUAL=ZMNG(ZELEM,ZCUR,WLBY2,WLBY2,RR)
04200      Z(POSPTR)=MUTUAL-IMAGE
04300      POSPTR=POSPTR+1
04400      XCUR=XCUR-DX
04500      ZCUR=ZCUR-DZ
04600 10    CONTINUE
04700      POSPTR=POSPTR+L-NT-NT+1
04800 20    CONTINUE
04900 30    READ(1,100) CODE,FLAG
05000      IF(CODE.EQ.1) CALL VCLTU(NCOLS,NROWS,V)
05100      IF(CODE.EQ.2) CALL VOLTC(NCOLS,NROWS,V)
05200      IF(CODE.EQ.3) CALL TAP(NCOLS,NROWS,DX,DZ,V)
05300      IF(CODE.EQ.4) CALL VOLTK(NCOLS,NROWS,V)
05400      IF(CODE.EQ.5) CALL STEER(NCOLS,NROWS,DX,DZ,V)
05500      IF(CODE.EQ.6) CALL PHASE(NCOLS,NROWS,V)
05600      CALL SOLVE(Z,V,Y,NCOLS,NROWS,NELEM,LM,MM,L,M,N,FXCITE)
05700      FXCITE=.FALSE.
05800      WRITE(3,300)
05900      READ(1,100) FLAG2
06000      IF(FLAG2.NE.0) GO TO 25

```



```

06100      CALL PATTRN(NCOLS,NROWS,NELEM,Y,DX,DZ)
06200      CALL SHAPE(V,NSIDE,NT)
06300      CALL PATTRN(NCOLS,NROWS,NELEM,V,DX,DZ)
06400      25  CONTINUE
06500      IF(FLAG.EQ.0) GO TO 30
06600      STOP
06700      100  FORMAT(2I0,5F0.0,I0)
06800      200  FORMAT(1H ,4E12.4)
06900      300  FORMAT(1H , 'TYPE 0 TO PRINT ARRAY FACTOR, ELSE 1')
07000      END
07100      C
07200      C  ROUTINE TO COMPUTE THE FAR FIELD PATTERN
07300      SUBROUTINE PATTRN(IO,MO,NO,Y,DX,DZ)
07400      INTEGER IO,MO,NO,PTR
07500      REAL DX,DZ,ALPHA,COSTHE,MAGE,DANG,T1,T2,T3,DEL
07600      COMPLEX Y(1),AF,XPHASE,ZPHASE,DXPH,DZPH,CMLX
07700      REAL PAT(361)
07800      WRITE(3,1000)
07900      READ(1,1100) NPTS,DANG
08000      ALPHA=0.0
08100      DO 300 II=1,NPTS
08200      COSTHE=COS(ALPHA*.17453293E-01)
08300      T3=DX*COSTHE
08400      T1=COS(T3)
08500      T2=SIN(T3)
08600      DXPH=CMLX(T1,T2)
08700      DZPH=DXPH
08800      AF=(0.0,0.0)
08900      ZPHASE=(1.0,0.0)
09000      DO 200 I=1,MO
09100      PTR=(I-1)*IO
09200      XPHASE=(1.0,0.0)
09300      DO 100 J=1,LO
09400      AF=AF+Y(PTR+J)*XPHASE*ZPHASE
09500      XPHASE=XPHASE*DXPH
09600      100  CONTINUE
09700      ZPHASE=ZPHASE*DZPH
09800      200  CONTINUE
09900      MAGE=CABS(AF)
10000      PAT(II)=MAGE
10100      WRITE(3,1200) ALPHA,MAGE,AF
10200      ALPHA=ALPHA+DANG
10300      300  CONTINUE
10400      WRITE(21,1300) (PAT(II),II=1,NPTS)
10500      1000  FORMAT(1H , 'NPTS AND ANGLE INCREMENT?')
10600      1100  FORMAT(10,E0.0)
10700      1200  FORMAT(1H ,F8.1,3E12.4)
10800      1300  FORMAT(8E11.4)
10900      RETURN
11000      END
11100      C
11200      C  ROUTINE TO ZEROIZE CORNERS
11300      SUBROUTINE SHAPE(Y,NS,LO)
11400      COMPLEX Y(1),CZERO
11500      CZERO=(0.0,0.0)
11600      DO 30 I=1,NS-1
11700      IPTR=(I-1)*LO
11800      DO 10 J=1,NS-I
11900      Y(IPTR+J)=CZERO
12000      10  CONTINUE

```



```
12100      DO 20 J=2*NS+I-1,3*NS-2
12200      Y(IPTR+J)=CZERO
12300 20    CCNTINUE
12400 30    CONTINUE
12500      DO 60 I=2*NS,3*NS-2
12600      IPTR=(I-1)*LO
12700      DO 40 J=1,I-2*NS+1
12800      Y(IPTR+J)=CZERO
12900 40    CCNTINUE
13000      DO 50 J=5*NS-I-2,3*NS-2
13100      Y(IPTR+J)=CZERO
13200 50    CONTINUE
13300 60    CONTINUE
13400      RETURN
13500      END
```

```

00050      SUBROUTINE SOLVE(A,B,LO,MO,NO,LM,MM,L,M,N,FXCITE)
00100  C      ROUTINE TO SOLVE THE MATRIX EQUATION A X = B
00150      LOGICAL FXCITE
00200      COMPLEX CTEMP,CZERO,A(1),X(4096),V(4096),B(1),Y(7396)
00300      INTEGER FLAG1,FLAG2,COUNT
00400      REAL CHNAVG,CHNMAX,CHANGE
00600      CZERO=(0.0,0.0)
01100  C      ZEROIZE Y AND V (EXPENDED B)
01300      DO 10 I=1,N
01400          V(I)=CZERO
01500      10  CONTINUE
01600      DO 20 I=1,NO
01700          Y(I)=CZERO
01800      20  CONTINUE
01900      IF(.NOT.FXCITE) GO TO 65
02000      DO 40 I=1,MO
02100          IPTR=(I-1)*L
02300          JPTR=IPTR+LO+LO-1
02400          DO 30 J=IPTR+1,IPTR+LO-1
02500              A(JPTR)=A(J)
02600              JPTR=JPTR-1
02700      30  CONTINUE
02800      40  CONTINUE
03000  C      FILL UP A ARRAY AND V ARRAY
03100      DO 60 I=1,MO-1
03200          IPTR=(I-1)*L
03300          JPTR=(MO+MO-I-1)*L
03400          DO 50 J=1,LO+LO-1
03500              A(JPTR+J)=A(IPTR+J)
03600      50  CONTINUE
03700      60  CONTINUE
03900      CALL TWODF(A,N,M,L,MM,LM)
04000      65  CONTINUE
04005      CALL ITER(A,B,X,V,Y,LO,MO,NO,LM,MM,L,M,N)
04010      RETURN
04015      END
04020      SUBROUTINE ITER(A,B,X,V,Y,LO,MO,NO,LM,MM,L,M,N)
04025      INTEGER COUNT,FLAG1,FLAG2
04030      COMPLEX CZERO,CTEMP,A(1),B(1),X(1),V(1),Y(1)
04035      REAL CHNAVG,CHNMAX,CHANGE
04040      COUNT=0
04045      CZERO=(0.0,0.0)
04100      70  JPTR=1
04200          COUNT=COUNT+1
04300          DO 90 I=1,MO
04400              IPTR=L*(MO-2+I)+LO
04500              DO 80 J=1,LO
04600                  V(IPTR)=B(JPTR)
04700                  JPTR=JPTR+1
04800                  IPTR=IPTR+1
04900      80  CONTINUE
05000      90  CONTINUE
05200  C      FIND V TRANSFORMED AND COMPUTE X TRANSFORMED
05300      CALL TWODF(V,N,M,L,MM,LM)
05400      DO 100 I=1,N
05500          X(I)=V(I)/A(I)
05600      100 CONTINUE
05700  C      GET X FROM X TRANSFORMED
05800      CALL ITWODF(X,N,M,L,MM,LM)
06100  C      TRUNCATE X AND SAVE X AFTER COMPUTING THE CONVERGENCE CRITERION

```

```

06200      CHNAVG=0.0
06300      CHNMAX=0.0
06400      DO 120 I=1,N
06500      IPTR=I/L
06600      JPTR=I-IPTR*L
06700      IF (JPTR.LE.LO.AND.JPTR.NE.0.AND.IPTR.LT.NO) GO TO 110
06800      X(I)=CZERO
06900      GO TO 120
07000      110 IPTR=IPTR*LO+JPTR
07100      CTEMP=X(I)
07200      CHANGE=CABS(CTEMP-Y(IPTR))/CABS(CTEMP)
07300      Y(IPTR)=CTEMP
07400      CHNAVG=CHNAVG+CHANGE
07500      IF (CHANGE.GT.CHNMAX) CHNMAX=CHANGE
07600      120 CONTINUE
07700      CHNAVG=(CHNAVG*100.0)/FLOAT(NO)
07800      CHNMAX=CHNMAX*100.0
07900      WRITE(3,1200) CHNAVG,CHNMAX,COUNT
08000      C      FIND THE TRANSFORM OF TRUNCATED X
08100      CALL TWODF(X,N,M,L,MM,LM)
08200      C      COMPUTE V TRANSFORMED
08300      DO 130 I=1,N
08400      V(I)=A(I)*X(I)
08500      130 CONTINUE
08600      C      GET V FROM V TRANSFORMED
08700      CALL ITWODF(V,N,M,L,MM,LM)
08800      C      COMPUTE THE ERROR CRITERION
08900      CHNAVG=0.0
09000      CHNMAX=0.0
09100      JPTR=1
09200      DO 150 I=1,NO
09300      IPTR=L*(NO-2+I)+LO
09400      DO 140 J=1,LO
09500      CTEMP=B(JPTR)
09600      CHANGE=CABS(CTEMP-V(IPTR))/CABS(CTEMP)
09700      CHNAVG=CHNAVG+CHANGE
09800      IF (CHANGE.GT.CHNMAX) CHNMAX=CHANGE
09900      IPTR=IPTR+1
10000      JPTR=JPTR+1
10100      140 CONTINUE
10200      150 CONTINUE
10300      C      ASK WHETHER OR NOT TO STOP AFTER REPORTING % FIELD ERROR
10400      CHNAVG=(CHNAVG*100.0)/FLOAT(NO)
10500      CHNMAX=CHNMAX*100.0
10600      WRITE(3,1300)CHNMAX,CHNAVG
10700      READ(1,1100)FLAG1
10800      IF (FLAG1.EQ.0) GO TO 70
10900      WRITE(3,1400) (Y(I),I=1,NO)
11000      C      ASK IF FIELD SHOULD BE PRINTED OUT ALSO
11100      WRITE(3,1500)
11200      READ(1,1100)FLAG2
11300      IF (FLAG2.NE.0) RETURN
11400      WRITE(3,1600) (V(I),I=1,N)
11500      RETURN
11600      1000 FORMAT(10E0.0)
11700      1100 FORMAT(4I0)
11800      1200 FORMAT(1H,'AVG CURRENT CHANGE=',E14.7,' %'/
11900      $          1H,'MAX CURRENT CHANGE=',E14.7,' %'/
12000      $          1H,'AFTER',I4,' ITERATIONS'/)
12100      1300 FORMAT(1H,'MAX FIELD ERROR = ',E15.7,' %'/

```

```
12200      $      1H , 'AVG FIELD ERROR = ', E15.7, ' %' /  
12300      $      1H , 'CONTINUE ITERATIONS? 0 FOR YES, 1 FOR NO, AND RETURN' / )  
12400 1400  FORMAT(1H , 'CURRENTS' // (1H , 10E11.4))  
12500 1500  FORMAT(1H , 'PRINT FIELDS? 0 FOR YES, 1 FOR NO, THEN RETURN' / )  
12600 1600  FORMAT(1H , 'RESULTANT FIELDS' // (1H , 10E11.4))  
12700      END
```

III. MAIN PROGRAM SEGMENT AND SUBROUTINES FOR THE PLANAR ARRAY PROBLEMS (THREE EXPANSION FUNCTIONS PER ANTENNA ELEMENT)

The difference between the main program segment for three expansion functions per antenna element and the main program segment for one expansion function per antenna element is that there are five distinct mutual impedance vectors \vec{Z} to be computed for the three expansion functions per antenna element solution. Therefore the computation of each \vec{Z} is done by the separate routine FILLZ. Similarly, MSOLVE and SOLVE routines differ mainly in that there are three distinct vectors each for the generalized voltage \vec{V} and \vec{J} to be computed by MSOLVE. SOLVE on the other hand, computes only one vector each of \vec{V} and \vec{J} .

Routine ZMNG computes the mutual impedance between two parallel segments of thin wire dipoles of same length which may or may not be offset from each other along one or more axes. SICI, VOLTU, VOLTC TAP, VOLTK, and PHASE are all from [9] and hence no description of them will be given here. FFT, IFFT, TWODF, and ITWODF are fast fourier transform and inverse transform routines for one and two dimensional discrete fourier transforms, respectively. PATT3E is the routine to compute the array factor from the current distribution solutions obtained by using three expansion functions per antenna element.

The last main program segment computes the array factors from the current distribution solutions obtained by using three expansion functions per antenna element and from the ideal solutions which ignore the mutual coupling between antenna elements.

```

00100 C
00200 C PROGRAM FOR TRIANGULAR PATTERN WITH THREE EXPANSIONS PER
00300 C ELEMENT
00400 LOGICAL FXCITE
00500 INTEGER CODE, FLAG, NRCWS, NCOLS, NELEM
00600 REAL TWOPI
00700 COMPLEX V2(484), ZA(4096), ZB(4096), ZC(4096), ZD(4096), ZE(4096)
00800 COMPLEX X1(4096), X2(4096), X3(4096)
00900 FXCITE=.TRUE.
01000 TWOPI=6.2831853
01100 CZERO=(0.0,0.0)
01150 OPEN(UNIT=21, FILE='MXCUR.DAT')
01200 READ(1,100) NSIDE, DX, DY, DZ, WLNTH, RAD
01300 NROWS=NSIDE*3-2
01400 NCOLS=NROWS
01500 CALL SIZE(NCOLS, NRCWS, LM, MM, L, M, N)
01600 DX=DX*TWOPI
01700 DY=DY*TWOPI
01800 DZ=DZ*TWOPI
01900 WLBY2=WLNTH*TWOPI/4.0
02000 ZELEM=WLBY2
02100 RAD=RAD*TWOPI
02200 NELEM=NROWS*NCOLS
02300 DY2SQ=4.0*DY*DY
02400 RAD2=RAD*RAD
02450 ZCFF=0.0
02500 CALL FILLZ(ZA, ZOFF, DX, DZ, DY2SQ, WLBY2, RAD2, NROWS, L, N)
02550 ZOFF=-ZELEM
02600 CALL FILLZ(ZB, ZOFF, DX, DZ, DY2SQ, WLBY2, RAD2, NRCWS, L, N)
02700 ZOFF=-ZELEM-ZELEM
02800 CALL FILLZ(ZC, ZOFF, DX, DZ, DY2SQ, WLBY2, RAD2, NRCWS, L, N)
02900 CALL FILLZ(ZD, ZELEM, DX, DZ, DY2SQ, WLBY2, RAD2, NROWS, L, N)
03000 ZOFF=ZELEM+ZELEM
03100 CALL FILLZ(ZE, ZCFF, DX, DZ, DY2SQ, WLBY2, RAD2, NRCWS, L, N)
03200 10 READ(1,200) CODE, FLAG
03300 IF(CODE.EQ.1) CALL VCLTU(NCOLS, NROWS, V2)
03400 IF(CODE.EQ.2) CALL VCLTC(NCOLS, NROWS, V2)
03500 IF(CODE.EQ.3) CALL TAP(NCOLS, NROWS, DX, DZ, V2)
03600 IF(CODE.EQ.4) CALL VOLTK(NCOLS, NROWS, V2)
03700 IF(CODE.EQ.5) CALL STEER(NCOLS, NROWS, DX, DZ, V2)
03800 IF(CODE.EQ.6) CALL PHASE(NCOLS, NROWS, V2)
03900 CALL MSOLVE(ZA, ZB, ZC, ZD, ZE, V2, X1, X2, X3, NCOLS, NROWS,
04000 $ NELEM, LM, MM, L, M, N, FXCITE)
04100 FXCITE=.FALSE.
04200 IF(FLAG.EQ.0) GO TO 10
04300 STOP
04400 100 FORMAT(I0,5F0.0,I0)
04450 200 FORMAT(2I0)
04500 END
04600 C
04700 C ROUTINE TO FILL THE Z MATRIX FOR TRIANGULAR ARRAYS
04800 SUBROUTINE FILLZ(Z, ZOFF, DX, DZ, DY2SQ, WLBY2, RAD2, NROWS, L, N)
04900 INTEGER POSPTR, NRCWS, L, N
05000 REAL DX, DZ, DY2SQ, ZELEM, XCUR, ZCUR, RR, WLBY2
05100 COMPLEX Z(1), MUTUAL, IMAGE, CZERO, ZMNG
05200 CZERO=(0.0,0.0)
05250 ZELEM=0.0
05300 PCSPTR=1
05400 DO 10 I=1, N
05500 Z(I)=CZERO

```



```

05600      10  CONTINUE
05700          NTOTAL=NROWS+NROWS-1
05800          DO 30 I=1,NTOTAL
05900              XCUR=(I-1)*DX
06000              ZCUR=(NTOTAL-I)*DZ+ZCFF
06100              DO 20 J=1,NTOTAL
06200                  RR=XCUR*XCUR+DY2SQ
06300                  IMAGE=ZMNG(ZELEM,ZCUR,WLEY2,RR)
06400                  RR=RR-DY2SQ
06500                  IF(RR.EQ.0.0) RR=RR+RAD2
06600                  MUTUAL=ZMNG(ZELEM,ZCUR,WLEY2,RR)
06700                  Z(POSPTR)=MUTUAL-IMAGE
06800                  POSPTR=PCSPTR+1
06900                  XCUR=XCUR-DX
07000                  ZCUR=ZCUR-DZ
07100      20  CONTINUE
07200          POSPTR=POSPTR+L-NTOTAL
07300      30  CONTINUE
07400          RETURN
07500          END
07600  C
07700  C      ROUTINE TO SOLVE THE CONVOLUTION EQUATIONS
07800          SUBROUTINE MSOLVE(ZA,ZB,ZC,ZD,ZE,B,X1,X2,X3,LO,MO,NO,
07900          $              LM,MM,L,M,N,FXCITE)
08000          INTEGER COUNT
08050          LOGICAL FXCITE
08100          COMPLEX CTEMP1,CTEMP2,CTEMP3,ZA(1),ZB(1),ZC(1),ZD(1),ZE(1),
08200          $          B(1),X1(1),X2(1),X3(1),V1(4096),V2(4096),V3(4096),
08300          $          T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,Y1(484),Y2(484),Y3(484),
08400          $          CZERO
08500          CZERC=(0.0,0.0)
08600          DO 10 I=1,NO
08700              Y1(I)=CZERO
08800              Y2(I)=CZERO
08900              Y3(I)=CZERO
09000      10  CONTINUE
09100          NS=(LC+2)/3
09200          EFAC=3.0*FLOAT(NO-2*(NS-1)*NS)
09300          DO 20 I=1,N
09400              V1(I)=CZERO
09500              V2(I)=CZERO
09600              V3(I)=CZERO
09700      20  CONTINUE
09800          IF(.NOT.FXCITE) GO TO 30
09900          CALL TWODF(ZA,N,M,L,MM,LM)
10000          CALL TWODF(ZB,N,M,L,MM,LM)
10100          CALL TWODF(ZC,N,M,L,MM,LM)
10200          CALL TWODF(ZD,N,M,L,MM,LM)
10300          CALL TWODF(ZE,N,M,L,MM,LM)
10400      30  ICENTR=L*(MO-1)+LC-1
10500          CCUNT=1
10600      40  CONTINUE
10800          DO 60 I=1,NS-1
10900              IPTR=(I-1)*L+ICENTR
11000              JPTR=(I-1)*LO
11100              DO 50 J=NS-I+1,2*NS+I-2
11200                  V1(IPTR+J)=CZERO
11300                  V2(IPTR+J)=B(JPTR+J)
11400                  V3(IPTR+J)=CZERO
11500      50  CONTINUE

```



```

11600      60  CONTINUE
11700      DO 80 I=NS,2*NS-1
11800      IPTR=(I-1)*L+ICENTR
11900      JPTR=(I-1)*LO
12000      DO 70 J=1,3*NS-2
12100      V1(IPTR+J)=CZERO
12200      V2(IPTR+J)=B(JPTR+J)
12300      V3(IPTR+J)=CZERO
12400      70  CONTINUE
12500      80  CONTINUE
12600      DO 100 I=2*NS,3*NS-2
12700      IPTR=(I-1)*L+ICENTR
12800      JPTR=(I-1)*LO
12900      DO 90 J=I+2-2*NS,5*NS-I-3
13000      V1(IPTR+J)=CZERO
13100      V2(IPTR+J)=B(JPTR+J)
13200      V3(IPTR+J)=CZERO
13300      90  CONTINUE
13400      100 CONTINUE
13450      105 CONTINUE
13500  C      FIND V1,V2,V3 TRANSFORMED AND COMPUTE X1,X2,X3
13600      CALL TWODF(V1,N,M,L,MM,LM)
13700      CALL TWODF(V2,N,M,L,MM,LM)
13800      CALL TWODF(V3,N,M,L,MM,LM)
13900      DO 110 I=1,N
14000      T1=ZD(I)/ZA(I)
14100      T2=ZE(I)/ZA(I)
14200      T3=ZA(I)-T1*ZB(I)
14300      T4=ZB(I)-T1*ZC(I)
14400      T5=ZA(I)-T2*ZC(I)
14500      T6=ZD(I)-T2*ZB(I)
14600      T7=V2(I)-V1(I)*T1
14700      T8=V3(I)-V1(I)*T2
14800      T9=T5-T4*T6/T3
14900      T10=T8-T7*T6/T3
15000      X3(I)=T10/T9
15100      X2(I)=(T7-T4*X3(I))/T3
15200      X1(I)=(V1(I)-ZB(I)*X2(I)-ZC(I)*X3(I))/ZA(I)
15300      110 CONTINUE
15400      CALL ITWODF(X1,N,M,L,MM,LM)
15500      CALL ITWODF(X2,N,M,L,MM,LM)
15600      CALL ITWODF(X3,N,M,L,MM,LM)
15700      DO 140 I=1,NS-1
15800      IPTR=(I-1)*L
15900      DO 120 J=1,NS-I
16000      X1(IPTR+J)=CZERO
16100      X2(IPTR+J)=CZERO
16200      X3(IPTR+J)=CZERO
16300      120 CONTINUE
16400      DO 130 J=2*NS+I-1,3*NS-2
16500      X1(IPTR+J)=CZERO
16600      X2(IPTR+J)=CZERO
16700      X3(IPTR+J)=CZERO
16800      130 CONTINUE
16900      140 CONTINUE
17000      DO 170 I=2*NS,3*NS-2
17100      IPTR=(I-1)*L
17200      DO 150 J=1,I-2*NS+1
17300      X1(IPTR+J)=CZERO
17400      X2(IPTR+J)=CZERO

```

```

17500      X3(IPTR+J)=CZERO
17600      150  CONTINUE
17700      DO 160 J=5*NS-I-2,3*NS-2
17800      X1(IPTR+J)=CZERO
17900      X2(IPTR+J)=CZERO
18000      X3(IPTR+J)=CZERO
18100      160  CONTINUE
18200      170  CONTINUE
18300  C
18400  C      COMPUTE THE CRITERIONS AND REPCBT
18500      CHNAVG=0.0
18600      CHNMAX=0.0
18700      DO 190 I=1,N
18800      IPTR=I/L
18900      JPTR=I-IPTR*L
19000      IF(JPTR.LE.LO.AND.JPTR.NE.0.AND.IPTR.IT.MC) GO TO 180
19100      X1(I)=CZERO
19200      X2(I)=CZERO
19300      X3(I)=CZERO
19400      GO TO 190
19500      180  CONTINUE
19600      IPTR=IPTR*LO+JPTR
19700      CTEMP1=X1(I)
19800      CTEMP2=X2(I)
19900      CTEMP3=X3(I)
20000      IF(CTEMP1.EQ.CZERO) GO TO 190
20100      CHNGE1=CABS(CTEMP1-Y1(IPTR))/CABS(CTEMP1)
20200      CHNGE2=CABS(CTEMP2-Y2(IPTR))/CABS(CTEMP2)
20300      CHNGE3=CABS(CTEMP3-Y3(IPTR))/CABS(CTEMP3)
20400      Y1(IPTR)=CTEMP1
20500      Y2(IPTR)=CTEMP2
20600      Y3(IPTR)=CTEMP3
20700      CHNAVG=CHNAVG+CHNGE1+CHNGE2+CHNGE3
20800      IF(CHNGE1.GT.CHNMAX) CHNMAX=CHNGE1
20900      IF(CHNGE2.GT.CHNMAX) CHNMAX=CHNGE2
21000      IF(CHNGE3.GT.CHNMAX) CHNMAX=CHNGE3
21100      190  CONTINUE
21200      CHNAVG=(CHNAVG*100.0)/EFACT
21300      CHNMAX=(CHNMAX*100.0)
21400      WRITE(3,1200) CHNAVG,CHNMAX,COUNT
21500  C
21600  C      GET THE TRANSFORM OF THE CURRENTS
21700      CALL TWODF(X1,N,M,L,MM,LM)
21800      CALL TWODF(X2,N,M,L,MM,LM)
21900      CALL TWODF(X3,N,M,L,MM,LM)
22000      DO 200 I=1,N
22100      V1(I)=ZA(I)*X1(I)+ZB(I)*X2(I)+ZC(I)*X3(I)
22200      V2(I)=ZD(I)*X1(I)+ZA(I)*X2(I)+ZB(I)*X3(I)
22300      V3(I)=ZE(I)*X1(I)+ZD(I)*X2(I)+ZA(I)*X3(I)
22400      200  CONTINUE
22500  C
22600  C      GET THE FIELD AND COMPUTE % ERRORS
22700      CALL ITWODF(V1,N,M,L,MM,LM)
22800      CALL ITWODF(V2,N,M,L,MM,LM)
22900      CALL ITWODF(V3,N,M,L,MM,LM)
23000      CHNAVG=0.0
23100      CHNMAX=0.0
23200      DO 220 I=1,NS-1
23300      IPTR=(I-1)*L+ICENTR
23400      JPTR=(I-1)*LO

```

```

23500      DO 210 J=NS-I+1,2*NS+I-2
23600      CTEMP1=B(JPTR+J)
23700      CHANGE=CABS(CTEMP1-V2(IPTR+J))/CABS(CTEMP1)
23800      IF(CHANGE.GT.CHNMAX) CHNMAX=CHANGE
23900      CHNAVG=CHNAVG+CHANGE
24000      V2(IPTR+J)=CTEMP1
24100      V1(IPTR+J)=CZERO
24200      V3(IPTR+J)=CZERO
24300      210 CONTINUE
24400      220 CONTINUE
24500      DO 240 I=NS,NS*2-1
24600      IPTR=(I-1)*L+ICENTR
24700      JPTR=(I-1)*LO
24800      DO 230 J=1,3*NS-2
24900      CTEMP1=B(JPTR+J)
25000      CHANGE=CABS(CTEMP1-V2(IPTR+J))/CABS(CTEMP1)
25200      IF(CHANGE.GT.CHNMAX) CHNMAX=CHANGE
25300      CHNAVG=CHNAVG+CHANGE
25400      V1(IPTR+J)=CZERO
25500      V2(IPTR+J)=CTEMP1
25600      V3(IPTR+J)=CZERO
25700      230 CONTINUE
25800      240 CONTINUE
25900      DO 260 I=2*NS,3*NS-2
26000      IPTR=(I-1)*L+ICENTR
26100      JPTR=(I-1)*LO
26200      DO 250 J=I+2-2*NS,5*NS-I-3
26300      CTEMP1=B(JPTR+J)
26400      CHANGE=CABS(CTEMP1-V2(IPTR+J))/CABS(CTEMP1)
26500      IF(CHANGE.GT.CHNMAX) CHNMAX=CHANGE
26600      CHNAVG=CHNAVG+CHANGE
26700      V2(IPTR+J)=CTEMP1
26800      V1(IPTR+J)=CZERO
26900      V3(IPTR+J)=CZERO
27000      250 CONTINUE
27100      260 CONTINUE
27200      C  ASK WHETHER OR NOT TO STOP AFTER REPORTING % FIELD ERROR
27300      CHNAVG=(CHNAVG*100.0)/EFACT
27400      CHNMAX=CHNMAX*100.0
27500      WRITE(3,1300)CHNAVG,CHNMAX
27600      READ(1,1100)FLAG1
27650      CCOUNT=COUNT+1
27700      IF(FLAG1.EQ.0) GO TO 105
27750      WRITE(3,1400)(Y1(I),I=1,NO)
27775      WRITE(21,1700)(Y1(I),I=1,NO)
27800      WRITE(3,1400)(Y2(I),I=1,NO)
27825      WRITE(21,1700)(Y2(I),I=1,NO)
27850      WRITE(3,1400)(Y3(I),I=1,NO)
27875      WRITE(21,1700)(Y3(I),I=1,NO)
27900      C  ASK IF FIELD SHOULD BE PRINTED OUT ALSO
28000      WRITE(3,1500)
28100      READ(1,1100) FLAG2
28200      IF(FLAG2.NE.0) RETURN
28300      WRITE(3,1600)(V2(I),I=1,N)
28400      RETURN
28500      1000 FORMAT(10E0.0)
28600      1100 FORMAT(4I0)
28700      1200 FORMAT(1H,'AVG CURRENT CHANGE=',E14.7,' %'/
28800      $          1H,'MAX CURRENT CHANGE=',E14.7,' %'//
28900      $          1H,'AFTER',I4,' ITERATIONS'//)

```

```

29000 1300 FORMAT(1H , 'AVG FIELD ERROR = ', E15.7, ' %' /
29100 $      1H , 'MAX FIELD ERROR = ', E15.7, ' %' /
29200 $      1H , 'CONTINUE ITERATIONS? 0 FOR YES, 1 FOR NO, AND RETURN' /)
29300 1400 FORMAT(1H , 'CURRENTS' // (1H , 10E11.4))
29400 1500 FORMAT(1H , 'PRINT FIELDS? 0 FOR YES, 1 FOR NO, THEN RETURN' /)
29500 1600 FORMAT(1H , 'RESULTANT FIELDS' // (1H , 10E11.4))
29550 1700 FORMAT(8E14.6)
29600 END
29700 C
29800 C COMPUTE MUTUAL IMPEDANCE BETWEEN TWO PARALLEL SEGMENTS
29900 C OF THIN WIRE DIPOLES OF SAME LENGTH
30000 FUNCTION ZMNG(Z1,Z2,LBY2,RSQ)
30100 REAL LBY2
30200 COMPLEX ZMNG,CMPLX
30300 DZ=ABS(Z1-Z2)
30400 CC=2.0*COS(LBY2)
30500 CSQ=CC*CC
30600 D1=DZ
30700 D2=DZ+LBY2
30800 D3=DZ-LBY2
30900 D4=D2+LBY2
31000 D5=D3-LBY2
31100 U1=SQRT(RSQ+D1*D1)+D1
31200 U2=SQRT(RSQ+D2*D2)+D2
31300 U3=SQRT(RSQ+D3*D3)+D3
31400 U4=SQRT(RSQ+D4*D4)+D4
31500 U5=SQRT(RSQ+D5*D5)+D5
31600 V1=RSQ/U1
31700 V2=RSQ/U2
31800 V3=RSQ/U3
31900 V4=RSQ/U4
32000 V5=RSQ/U5
32100 CALL SICI(SU1,CU1,U1)
32200 CALL SICI(SU2,CU2,U2)
32300 CALL SICI(SU3,CU3,U3)
32400 CALL SICI(SU4,CU4,U4)
32500 CALL SICI(SU5,CU5,U5)
32600 CALL SICI(SV1,CV1,V1)
32700 CALL SICI(SV2,CV2,V2)
32800 CALL SICI(SV3,CV3,V3)
32900 CALL SICI(SV4,CV4,V4)
33000 CALL SICI(SV5,CV5,V5)
33100 S1=SIN(D1)
33200 S2=SIN(D2)
33300 S3=SIN(D3)
33400 S4=SIN(D4)
33500 S5=SIN(D5)
33600 C1=COS(D1)
33700 C2=COS(D2)
33800 C3=COS(D3)
33900 C4=COS(D4)
34000 C5=COS(D5)
34100 RL=(2.0+CSQ)*(C1*(CU1+CV1)+S1*(SU1-SV1))
34200 $      -2.0*CC*(C2*(CU2+CV2)+S2*(SU2-SV2)+C3*(CU3+CV3)
34300 $      +S3*(SU3-SV3))+C4*(CU4+CV4)+S4*(SU4-SV4)
34400 $      +C5*(CU5+CV5)+S5*(SU5-SV5)
34500 AG=(2.0+CSQ)*(S1*(CU1-CV1)-C1*(SU1+SV1))
34600 $      -2.0*CC*(S2*(CU2-CV2)-C2*(SU2+SV2)+S3*(CU3-CV3)
34700 $      -C3*(SU3+SV3))+S4*(CU4-CV4)-C4*(SU4+SV4)
34800 $      +S5*(CU5-CV5)-C5*(SU5+SV5)

```

```

34900      ZMNG=15.0*CMPLX(RI,AG)/(SIN(LBY2)*SIN(LBY2))
35000      RETURN
35100      END
35600      C
35700      C
35800      SUBROUTINE SICI(SI,CI,X)
35900      Z=ABS(X)
36000      IF(Z-4.) 1,1,4
36100      1 Y=(4.-Z)*(4.+Z)
36200      3 SI=X*(((1.753141E-9*Y+1.568988E-7)*Y+1.374168E-5)*Y+6.939889E-4)
36300      1*Y+1.964882E-2)*Y+4.395509E-1)
36400      CI=((5.772156E-1+ALOG(Z))/Z-Z*(((1.386985E-10*Y+1.584996E-8)*Y
36500      1+1.725752E-6)*Y+1.185999E-4)*Y+4.990920E-3)*Y+1.315308E-1))*Z
36600      RETURN
36700      4 SI=SIN(Z)
36800      Y=COS(Z)
36900      Z=4./Z
37000      U=(((((((4.048069E-3*Z-2.279143E-2)*Z+5.515070E-2)*Z-7.261642E-2)
37100      1*Z+4.987716E-2)*Z-3.332519E-3)*Z-2.314617E-2)*Z-1.134958E-5)*Z
37200      2+6.250011E-2)*Z+2.583989E-10
37300      V=((((((((-5.108699E-3*Z+2.819179E-2)*Z-6.537283E-2)*Z
37400      1+7.902034E-2)*Z-4.400416E-2)*Z-7.945556E-3)*Z+2.601293E-2)*Z
37500      2-3.764000E-4)*Z-3.122418E-2)*Z-6.646441E-7)*Z+2.500000E-1
37600      CI=Z*(SI*V-Y*U)
37700      SI=-Z*(SI*U+Y*V)      +1.570796
37800      RETURN
37900      END
38000      C
38100      C
38200      C
38300      SUBROUTINE VOLTU(M2,M3,V)
38400      C      UNIFORM EXCITATION WITH SPECIFIED AMPLITUDE AND PHASE
38500      COMPLEX V(1),CMPLX,VALUE
38600      M23=M2*M3
38700      READ (5,1)AM,PH
38800      1  FORMAT(2F0.0)
38900      RAD=PH*3.14159/180.
39000      VALUE=CMPLX(AM*COS(RAD),AM*SIN(RAD))
39100      DO 2 I=1,M23
39200      V(I)=VALUE
39300      2  CONTINUE
39400      WRITE(5,3)AM,PH
39500      3  FORMAT(///1X,'UNIFORM VOLTAGE EXCITATION -'//1X,
39600      8'MAGNITUDE =' ,1P1E20.5,' PHASE =' ,1P1E20.5/)
39700      RETURN
39800      END
39900      C
40000      C
40100      C
40200      SUBROUTINE VOLTC(M2,M3,V)
40300      C      READ IN COMPLEX NUMBERS AS VOLTAGE FOR EACH DIPOLE
40400      COMPLEX V(1)
40500      M23=M2*M3
40600      READ (5,1) (V(I),I=1,M23)
40700      1  FORMAT(2F0.0)
40800      WRITE(5,2)
40900      2  FORMAT(///1X,'ARBITRARY VOLTAGE EXCITATION -'//)
41000      RETURN
41100      END
41200      C

```



```

41300 C
41400 SUBROUTINE STEER(M2,M3,DX,DZ,PP)
41500 C PROGRESSIVE PHASE SHIFT ON EACH DIPOLE -
41600 C STEERING THE MAIN BEAM IN BOTH DIRECTION
41700 COMPLEX PP(1),CMPLX
41800 READ (5,1) RZ,RX
41900 1 FORMAT(2F0.0)
42000 PI2=6.2831853
42100 THX=(RX)*PI2/360.
42200 THZ=(RZ)*PI2/360.
42300 L=0
42400 XK=-DX*COS(THX)
42500 ZK=-DZ*COS(THZ)
42600 DO 100 I=1,M3
42700 DO 100 J=1,M2
42800 L=L+1
42900 PH=FLOAT(J-1)*XK+FLOAT(I-1)*ZK
43000 PP(L)=1.0*CMPLX(COS(PH),SIN(PH))
43100 100 CONTINUE
43200 WRITE(5,20)RX,RZ
43300 20 FORMAT(///1X,'BEAM STEERING =',F10.5,' DEGREES IN PHI ANGLE',
43400 &/1X,14X,'=',F10.5,' DEGREES IN THE ANGLE'//)
43500 RETURN
43600 END
43700 C
43800 C
43900 SUBROUTINE TAP(M2,M3,EX,DZ,VV)
44000 C MAGNITUDE TAPER OF EXCITATION IN BOTH DIRECTION
44100 COMPLEX VV(1)
44200 PI2=6.2831853
44300 HFX=DX*(M2-1)*0.5
44400 HFZ=DZ*(M3-1)*0.5
44500 L=0
44600 WRITE(5,2)
44700 2 FORMAT(///1X,'EXPONENTIAL TAPERED IN MAGNITUDE'///)
44800 DO 100 I=1,M3
44900 Z=((I-1)*DZ-HFZ)/PI2
45000 FUNZ=EXP(-ABS(Z))
45100 DO 100 J=1,M2
45200 L=L+1
45300 X=((J-1)*DX-HFX)/PI2
45400 VV(L)=EXP(-ABS(X))*FUNZ
45500 100 CONTINUE
45600 RETURN
45700 END
45800 C
45900 C
46000 C
46100 SUBROUTINE VOLTK(M2,M3,V)
46200 COMPLEX V(1),VK(20),VJ(20)
46300 READ (5,1) (VK(I),I=1,M3)
46400 READ (5,1) (VJ(I),I=1,M2)
46500 1 FORMAT(10F0.0)
46600 L=0
46700 DO 2 I=1,M3
46800 DO 2 J=1,M2
46900 L=L+1
47000 V(L)=VK(I)*VJ(J)
47100 2 CONTINUE
47200 WRITE(5,4)

```

```

47300      4      FORMAT (///1X,'VOLTAGE EXCITATION - SPECIFIED BY ROW',
47400      &' AND COLUMN',///)
47500      RETURN
47600      END
47700      C
47800      C
47900      SUBROUTINE PHASE(M2,M3,PP)
48000      C      PROGRESSIVE PHASE SHJFT CN EACH DIPOLE -
48100      COMPLEX PP(1),CMPLX
48200      READ (5,1)RZ,RX
48300      1      FORMAT(2F0.0)
48400      PI2=6.2831853
48500      THX=(RX)*PI2/360.
48600      THZ=(RZ)*PI2/360.
48700      L=0
48800      DO 100 I=1,M3
48900      DO 100 J=1,M2
49000      L=L+1
49100      PH=FLOAT(J-1)*THX+FLOAT(I-1)*THZ
49200      PP(L)=1.0*CMPLX(COS(PH),SIN(PH))
49300      100 CONTINUE
49400      WRITE(5,20)RX,RZ
49500      20      FORMAT(///1X,'PROGRESSIVE PHASE SHIFT = ',F10.5,
49600      &'DEGREES IN ROW DIRECTION'/25X,'= ',F10.5,
49700      &'DEGREES IN COLUMN DIRECTION',///)
49800      RETURN
49900      END
50000      SUBROUTINE SIZE(LO,MO,LM,MM,L,M,N)
50100      L=LO*3-2
50200      M=MO*3-2
50300      LTEMP=L
50400      MTEMP=M
50500      LM=0
50600      MM=0
50700      1      L=L/2
50800      LM=LM+1
50900      IF(L.GT.1) GO TO 1
51000      2      M=M/2
51100      MM=MM+1
51200      IF(M.GT.1) GO TO 2
51300      L=2**LM
51400      M=2**MM
51500      IF(LTEMP.GT.L) LM=LM+1
51600      IF(LTEMP.GT.L) L=L*2
51700      IF(MTEMP.GT.M) MM=MM+1
51800      IF(MTEMP.GT.M) M=M*2
51900      N=M*L
52000      RETURN
52100      END
52200      SUBROUTINE FFT(X,M,START,STEP)
52300      COMPLEX X(16384),U,W,T
52400      INTEGER START,STEP,SCIFF
52500      N=2**M
52600      SCIFF=STEP-START
52700      NV2 =N/2*STEP
52800      NM1 =(N-2)*STEP+START
52900      N    =(N-1)*STEP+START
53000      J    =START
53100      DO 8 I=START,NM1,STEP
53200      IF(I.GE.J) GO TO 5

```

```

53300      T      =X(J)
53400      X(J) =X(I)
53500      X(I) =T
53600      5      K      =NV2
53700      6      IF (K-SDIFF.GE.J) GC TO 7
53800      J      =J-K
53900      K      =K/2
54000      GC TO 6
54100      7      J      =J+K
54200      8      CONTINUE
54300      PI      =3.14159265358979
54400      DO 20 L=1,M
54500      LE      =2**L
54600      LE1=LE/2
54700      LSTEP=LE1*STEP
54800      U      =(1.0,0.0)
54900      ANGLE=PI/FLOAT(LE1)
55000      W      =CMPLX(COS(ANGLE),-SIN(ANGLE))
55100      LE1 =LSTEP+START-STEP
55200      LE      =LE*STEP
55300      DO 20 J=START,LE1,STEP
55400      DO 10 I=J,N,LE
55500      IP      =I+LSTEP
55600      T      =X(IP)*U
55700      X(IP)=X(I)-T
55800      X(I) =X(I)+T
55900      10     CONTINUE
56000      U=U*W
56100      20     CONTINUE
56200      RETURN
56300      END
56400  C
56500      SUBROUTINE TWODF(X,N,M,L,MM,LM)
56600      COMPLEX X(16384)
56700      INTEGER START,STEP
56800      START=1
56900      STEP =1
57000      DO 10 I=1,M
57100      CALL FFT(X,LM,START,STEP)
57200      START=START+L
57300      10     CONTINUE
57400      STEP =L
57500      DO 20 I=1,L
57600      START=I
57700      CALL FFT(X,MM,START,STEP)
57800      20     CONTINUE
57900      RETURN
58000      END
58100      SUBROUTINE IFFT(X,M,START,STEP)
58200      COMPLEX X(16384),U,W,T
58300      INTEGER START,STEP,SDIFF
58400      N=2**M
58500      SDIFF=STEP-START
58600      NW2 =N/2*STEP
58700      NM1 =(N-2)*STEP+START
58800      NEXP =(N-1)*STEP+START
58900      J      =START
59000      DO 8 I=START,NM1,STEP
59100      IF(I.GE.J) GC TO 5
59200      T      =X(J)

```



```

59300      X(J) =X(I)
59400      X(I) =T
59500      5   K      =NV2
59600      6   IF(K-SDIFF.GE.J) GC TO 7
59700      J      =J-K
59800      K      =K/2
59900      GO TO 6
60000      7   J      =J+K
60100      8   CONTINUE
60200      PI      =3.14159265358979
60300      DO 20 L=1,M
60400      LE      =2**L
60500      LE1=LE/2
60600      LSTEP=LE1*STEP
60700      U      =(1.0,0.0)
60800      ANGLE=PI/FLOAT(LE1)
60900      W      =CMPLX(COS(ANGLE),SIN(ANGLE))
61000      LE1 =LSTEP+START-STEP
61100      LE      =LE*STEP
61200      DO 20 J=START,LE1,STEP
61300      DO 10 I=J,NEXP,LE
61400      IP      =I+LSTEP
61500      T      =X(IP)*U
61600      X(IP)=X(I)-T
61700      X(I) =X(I)+T
61800      10  CONTINUE
61900      U=U*W
62000      20  CONTINUE
62100      RETURN
62200      END
C
62300      SUBROUTINE ITWOEF(X,N,M,L,MM,LM)
62400      COMPLEX X(16384)
62500      INTEGER START,STEP
62600      START=1
62700      STEP =1
62800      DO 10 I=1,M
62900      CALL IFFT(X,LM,START,STEP)
63000      START=START+L
63100      10  CONTINUE
63200      STEP =L
63300      DO 20 I=1,L
63400      START=I
63500      CALL IFFT(X,MM,START,STEP)
63600      20  CONTINUE
63700      FN=FLOAT(N)
63800      DO 30 I=1,N
63900      X(I)=X(I)/FN
64000      30  CONTINUE
64100      RETURN
64200      END
64300

```

```

00100 C
00200 C      ROUTINE TO COMPUTE THE FAR FIELD PATTERN
00300      SUBROUTINE PATTRN(LO,MO,NO,Y,DX,DZ)
00400      INTEGER LO,MO,NO,PTR
00500      REAL DX,DZ,ALPHA,COSTHE,MAGE,DANG,T1,T2,T3,T4,DEL
00600      COMPLEX Y(1),AF,XPHASE,ZPHASE,DXPH,DZPH,CMPLX
00700      REAL PAT(361)
00800      WRITE(3,1000)
00900      READ(1,1100) NPTS,DANG,IFLAG
01000      ALPHA=0.0
01100      DO 300 II=1,NPTS
01200      COSTHE=COS(ALPHA*.17453293E-01)
01300      T3=DX*COSTHE
01400      T1=COS(T3)
01500      T2=SIN(T3)
01550      T4=-T2
01600      DXPH=CMPLX(T1,T2)
01700      DZPH=DXPH
01800      AF=(0.0,0.0)
01900      ZPHASE=(1.0,0.0)
01950      IF(IFLAG.NE.0) DZPH=CMPLX(T1,T4)
02000      DO 200 I=1,MO
02100      PTR=(I-1)*LO
02200      XPHASE=(1.0,0.0)
02300      DO 100 J=1,LO
02400      AF=AF+Y(PTR+J)*XPHASE*ZPHASE
02500      XPHASE=XPHASE*DXPH
02600      100  CONTINUE
02700      ZPHASE=ZPHASE*DZPH
02800      200  CONTINUE
02900      MAGE=CABS(AF)
03000      PAT(II)=MAGE
03100      WRITE(3,1200) ALPHA,MAGE,AF
03200      ALPHA=ALPHA+DANG
03300      300  CONTINUE
03400      WRITE(21,1300) (PAT(II),II=1,NPTS)
03500      1000  FORMAT(1H,'NPTS AND ANGLE INCREMENT?')
03600      1100  FORMAT(I0,E0.0,I0)
03700      1200  FORMAT(1H,'F8.1,3E12.4)
03800      1300  FORMAT(8E11.4)
03900      RETURN
04000      END

```

```

04100 C
04200 C      ROUTINE TO ZEROIZE CORNERS
04300      SUBROUTINE SHAPE(Y,NS,LO)
04400      COMPLEX Y(1),CZERO
04500      CZERO=(0.0,0.0)
04600      DO 30 I=1,NS-1
04700      IPTR=(I-1)*LO
04800      DO 10 J=1,NS-I
04900      Y(IPTR+J)=CZERO
05000      10  CONTINUE
05100      DO 20 J=2*NS+I-1,3*NS-2
05200      Y(IPTR+J)=CZERO
05300      20  CONTINUE
05400      30  CONTINUE
05500      DO 60 I=2*NS,3*NS-2
05600      IPTR=(I-1)*LO
05700      DO 40 J=1,I-2*NS+1
05800      Y(IPTR+J)=CZERO

```

```

05900      40  CONTINUE
06000      DO 50 J=5*NS-I-2,3*NS-2
06100      Y(IPTR+J)=CZERO
06200      50  CONTINUE
06300      60  CONTINUE
06400      RETURN
06500      END
06600  C
06700  C      ROUTINE TO COMPUTE THE FAR FIELD PATTERN(3 EXPANSIONS)
06800      SUBROUTINE PATT3E(LO,MO,NO,Y,DX,DZ,WLBY2)
06900      INTEGER LO,MO,NO,PTR,CPTR
07000      REAL DX,DZ,ALPHA,COSTHE,MAGE,DANG,T1,T2,T3,T4,DEL
07100      COMPLEX Y(1),AF1,AF2,AF3,XPHASE,ZPHASE,DXPH,DZPH,CMPLX,AF
07200      COMPLEX PHASE,FAC1,FAC3
07300      REAL PAT(361)
07400      WRITE(3,1000)
07500      READ(1,1100) NPTS,DANG,IFLAG
07600      ALPHA=0.0
07700      DO 300 II=1,NPTS
07800      COSTHE=COS(ALPHA*.17453293E-01)
07900      T3=DX*COSTHE
08000      T1=COS(T3)
08100      T2=SIN(T3)
08150      T4=-T2
08200      DXPH=CMPLX(T1,T2)
08300      DZPH=DXPH
08400      AF1=(0.0,0.0)
08500      AF2=(0.0,0.0)
08600      AF3=(0.0,0.0)
08700      ZPHASE=(1.0,0.0)
08750      IF(IFLAG.NE.0) DZPH=CMPLX(T1,T4)
08800      DO 200 I=1,MO
08900      PTR=(I-1)*LO
09000      XPHASE=(1.0,0.0)
09100      DO 100 J=1,LO
09200      PHASE=XPHASE*ZPHASE
09300      CPTR=PTR+J
09400      AF1=AF1+Y(CPTR)*PHASE
09500      CPTR=CPTR+NO
09600      AF2=AF2+Y(CPTR)*PHASE
09700      CPTR=CPTR+NO
09800      AF3=AF3+Y(CPTR)*PHASE
09900      XPHASE=XPHASE*DXPH
10000      100  CONTINUE
10100      ZPHASE=ZPHASE*DZPH
10200      200  CONTINUE
10250      AF=AF1+AF2+AF3
10300      T3=WLBY2/2.0*COSTHE
10400      T1=COS(T3)
10500      T2=SIN(T3)
10600      FAC1=CMPLX(T1,T2)
10700      T2=-T2
10800      FAC3=CMPLX(T1,T2)
10850      IF(IFLAG.EQ.0) GO TO 250
10900      AF=AF1*FAC1+AF2+AF3*FAC3
10925      IF(COSTHE.GE.0.99999) GO TO 250
10950      AF=AF*(T1-COS(WLBY2/2.0))/(COS(WLBY2*COSTHE)-COS(WLBY2))
10975      250  CONTINUE
11000      MAGE=CABS(AF)
11100      PAT(II)=MAGE

```

```

11200      WRITE(3,1200) ALPHA,MAGE,AF
11300      ALPHA=ALPHA+DANG
11400      300  CONTINUE
11500      WRITE(21,1300) (PAT(II),II=1,NPTS)
11600      1000 FORMAT(1H,'NPTS AND ANGLE INCREMENT?')
11700      1100 FORMAT(I0,E0.0,I0)
11800      1200 FORMAT(1H,F8.1,3E12.4)
11900      1300 FORMAT(8E11.4)
12000      RETURN
12100      END
12200  C
12300  C      MAIN PROGRAM
12400      COMPLEX V(12300)
12450      INTEGER CODE,FLAG,LO,MO,NO,SKIP
12500      OPEN(UNIT=21,FILE='PATTRN.DAT')
12600      OPEN(UNIT=22,FILE='MXCUR.DAT')
12605      READ(1,200) LO,MO,NO,DX,DZ,WLBY2,SKIP
12610      10  IF(SKIP.LE.0) GO TO 20
12620      READ(22,300) (V(I),I=1,NO)
12630      READ(22,300) (V(I),I=1,NO)
12640      READ(22,300) (V(I),I=1,NO)
12650      SKIP=SKIP-1
12660      GO TO 10
12670      20  CONTINUE
12720      TWOPI=6.2831853
12760      DX=DX*TWOPI
12800      DZ=DZ*TWOPI
12820      WLBY2=WLBY2*TWOPI
12900      READ(22,300) (V(I),I=1,NO)
12930      READ(22,300) (V(I),I=NO+1,NO+NO)
12960      READ(22,300) (V(I),I=NO+NO+1,NO+NO+NO)
13000      CALL PATT3E(LO,MO,NO,V,DX,DZ,WLBY2)
13100      NSIDE=(LO+2)/3
13130      NCOLS=LO
13160      NROWS=MO
13200      READ(1,200) CODE,FLAG
13300      IF(CODE.EQ.1) CALL VOLTU(NCOLS,NROWS,V)
13400      IF(CODE.EQ.2) CALL VOLTC(NCOLS,NROWS,V)
13500      IF(CODE.EQ.3) CALL TAP(NCOLS,NROWS,DX,DZ,V)
13600      IF(CODE.EQ.4) CALL VOLTK(NCOLS,NROWS,V)
13700      IF(CODE.EQ.5) CALL STEER(NCOLS,NROWS,DX,DZ,V)
13800      IF(CODE.EQ.6) CALL PHASE(NCOLS,NROWS,V)
13900      CALL SHAPE(V,NSIDE,LO)
14000      CALL PATTRN(LO,MO,NO,V,DX,DZ)
14100      STOP
14200      200  FORMAT(3I0,3F0.0,I0)
14300      300  FORMAT(8E14.6)
14400      END

```

REFERENCES

- [1] H. L. Nyo and R. F. Harrington, "The Discrete Convolution Method for Solving some Large Moment Matrix Equations," Technical Report No. 21 on Grant No. N00014-76-0025 with the Department of Navy, Office of Naval Research, July 1983.
- [2] W. L. Ko and R. Mittra, "A Method for Combining Integral Equation and Asymptotic Techniques for Solving Electromagnetic Scattering Problems," Technical Report 76-6 on Grant No. DAHCO4-74-G-0113 with U.S. Army Research Office and Grant No. N00014-75-C-0293 with Office of Naval Research, May 1976.
- [3] R. Mittra, Y. Rahmat-Samii, and W. L. Ko, "Spectral Theory of Diffraction," Appl. Phys., vol. 10, pp. 1-13, 1976.
- [4] Y. Rahmat-Samii and R. Mittra, "Spectral Analysis of High Frequency Diffraction of an Arbitrary Incident field by a Half Plane -- Comparison with four Asymptotic Techniques," Rad. Sci., vol. 13, pp. 31-48, 1978.
- [5] R. Kastner and R. Mittra, "A Spectral-Interaction Technique for Analyzing Scattering from Arbitrary Bodies, Part I: Conducting Cylinders with E-wave Incidence," IEEE Trans. Antennas and Propagat., vol. AP-31, No. 3, pp. 499-506, May 1983.
- [6] A. V. Oppenheim and R. W. Schaffer, "Digital Signal Processing," Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1975.

- [7] R. F. Harrington, "Field Computation by Moment Methods," The Macmillan Company, New York, 1968.
- [8] D. C. Kuo and B. J. Strait, "Improved Programs for Analysis of Radiation and Scattering by Configurations of Arbitrarily Bent Thin Wires," Scientific Report No. 15 on Contract No. F19628-68-C-0180 with Air Force Cambridge Research Laboratories, Report AFCRL-72-0051, January 1972.
- [9] E. Ngai and A. T. Adams, "A Computer Program for a Planar Array of Thin Wire Dipoles Parallel to a Perfectly Conducting Plane," Scientific Report TR-80-3, Syracuse University, March 1980.